
Article:
Gleirscher, Mario orcid.org/0000-0002-9445-6863, Calinescu, Radu orcid.org/0000-0002-2678-9260 and Woodcock, Jim orcid.org/0000-0001-7955-2702 (2021) RiskStructures : A Design Algebra for Risk-Aware Machines. Formal Aspects of Computing. pp. 763-802. ISSN 1433-299X

https://doi.org/10.1007/s00165-021-00545-4

---

https://doi.org/10.1007/s00165-021-00545-4

Formal Aspects of Computing

Formal Aspects of Computing

RISKSTRUCTURES: A design algebra for risk-aware machines

Mario Gleirscher [1,2] Radu Calinescu [2,3] and Jim Woodcock

Computer Science Department, University of York, York, UK

Abstract. Machines, such as mobile robots and delivery drones, incorporate controllers responsible for a task *while handling risk* (e.g. anticipating and mitigating hazards; preventing and alleviating accidents). We refer to machines with this capability as risk-aware machines. *Risk awareness* includes robustness and resilience and complicates monitoring (i.e., introspection, sensing, prediction), decision making, and control. From an engineering perspective, risk awareness adds a range of dependability requirements to *system assurance*. Such assurance mandates a correct-by-construction approach to controller design, based on mathematical theory. We introduce RISKSTRUCTURES, an algebraic framework for risk modelling intended to support the design of *safety controllers* for risk-aware machines. Using the concept of a *risk factor* as a modelling primitive, this framework provides facilities to construct, examine, and assure these controllers. We prove desirable algebraic properties of these facilities, and demonstrate their applicability by using them to specify key aspects of safety controllers for risk-aware automated driving and collaborative robots.

Keywords: Correct construction; Formal development; Risk awareness; Run-time mitigation; Safety controllers; Robots and autonomous systems

1. Introduction

Humans identify and predict dangers and take actions to avoid or recover from dangerous situations. Likewise, autonomous machines can be expected to handle *risk* (e.g. hazardous environmental situations, erroneous control inputs, machine faults) on their own. Such machines should continuously judge risk in a situation, using introspection, estimating the current situation, and predicting future situations. While some risks can be avoided, for most of them only the *likelihood of their occurrence* or the *likelihood and severity of their consequences* can be reduced. This circumstance requires a careful analysis of risk causes and consequences in terms of *risk factors*, and the investigation of temporal and causal relationships between these. Risk can be handled by various measures [HM99, AASB+06], ranging from passive measures to *active control schemes*. The former include, for example, flexible surfaces or soft materials used on robot arms and predetermined breaking points in machine chassis [HASH09]. The latter include low-level continuous control mechanisms [SC88], high-leveldiscrete-event

Correspondence to: Mario Gleirscher, e-mail: mario.gleirscher@york.ac.uk

[1]Mario Gleirscher was supported in part by the German Research Foundation (DFG) under the Fellowship Grant no. 381212925.

[2]Work by Radu Calinescu and Mario Gleirscher was partially supported by the Lloyd's Register Foundation under the Autonomy Assurance International Programme (AAIP) Grant CSI:Cobot.

[3]Radu Calinescu was additionally supported by the UKRI Project EP/V026747/1 "Trustworthy Autonomous Systems Node in Resilience".

Published online: 26 May 2021

controllers [MGW+18, Gle17], and remote or supervisory control schemes [She03]. Overall, active control schemes detect violations of safety properties and try to reestablish a safe state.

Objective and hypothesis Decades ago, it was suggested that machine autonomy requires extensive sensor fusion, qualitative reasoning (e.g. [HM99]), and the internalisation of models of consciousness [HG03]. However, to this date, domains such as human-robot collaboration and autonomous driving are far from full autonomy in unrestricted environments, to a good extent because of difficulties in *assurance* [KW17, AZI+17]. As similar engineering challenges have been solved through formal design techniques (see, e.g. the survey papers [GM20, GFW19]), we hypothesise that these difficulties can be reduced by *bespoke rigorous design techniques*[4] beyond what can be achieved by employing general-purpose formal design methods alone. Such techniques could guide the correct construction of *safety controllers* that, inspired by [HM99, HG03], integrate a simple form of consciousness of risk, hereafter called *risk awareness*, into a machine.

Contributions Based on this hypothesis, we introduce RiskStructures, an algebraic framework for designing correct-by-construction safety controllers for risk-aware machines. In line with conventional risk analysis, this framework uses the notion of a *risk factor* as a modelling primitive and, going beyond preliminary results from our previous work [Gle17, Gle18], enhances its formalisation and proves key algebraic properties of a factor-based design calculus for risk-aware controllers. Our work focuses on qualitative risk modelling, enabling the use of different approaches to dealing with uncertainty. A risk structure can be used to formally specify *acceptance criteria*[5] and from these, one can synthesise safety controllers [GC20] and reason about their correctness. The presented theory is supported by the research tool Yap [Gle20], which interprets risk structures encoded in a domain-specific language and automates analysis and synthesis steps. To the best of our knowledge, this work is the first to give an algebraic account of risk modelling while building a bridge to safety controller design for risk-aware machines, adding design guidance to state-of-the-art dependability and risk assessment techniques, monitoring approaches, and optimal control models (Sect. 8).

The remainder of this article is structured as follows. After summarising background material and the used formalism (Sect. 2), Sects. 3 to 7 present our framework, using examples (Sect. 3 and sects. 4.3, 6.2 and 7.4) and discussing the rationale for its components (Sects. 5.5 and 6.3). Sect. 7.6 summarises RiskStructures as an approach to model uncertain negative outcomes of a machine's actions. Based on [BFK13], Table 5 highlights how low-level uncertainties can be dealt with in RiskStructures. After a comparison to previous research in Sect. 8, we add concluding remarks in Sect. 9. Proof details are listed in Appendix A.

2. Preliminaries

Beyond the avoidance and removal of development mistakes [ALRL04, Sec. 3.3.2] by rigorous design [GFW19] and dependability techniques [ALRL04, Sec. 5.3.1], there are *immanent operational risks* to be dealt with by bespoke machine features. This section introduces terminology, gives an overview of risk analysis and handling for robots and autonomous systems, and explains the formalism used in this work.

2.1. Risk analysis, assessment, and handling

Risk is the possibility of *undesired outcomes* (e.g. loss, injury, damage) of an action (whether nominal, faulty, or malicious) with *uncertain alternative outcomes* [KG81].[6] Particularly relevant are the actions (e.g. failures, incidents, accidents) in hazardous states (e.g. faults). Risk can be assessed by the *likelihood of a hazard*, the *likelihood of exposure* of an asset to this hazard, the *likelihood of an undesired outcome* if this hazard occurs, and the *severity* of this outcome [Lev95, Kum07]. As such, it is useful to identify causal relationships between events and to quantify risk using stochastic models (cf. reliability analysis of repairable systems [Kum07]) .

[4] That is, techniques dedicated to capturing risk factors.

[5] For readers familiar with refinement-based development: a risk structure $\mathfrak{R}$ refined by a process P, after hiding P's irrelevant events, can be used to express risk acceptance in P and verify this criterion of P.

[6] For the sake of simplicity, we neglect for whom the outcome is undesired and who performs the action. However, such aspects are useful and can be added.

Knowledge about risk often stems from accidents [SR02]. Methods such as *hazard and operability studies* (HazOp), event tree analysis, and layer of protection analysis help engineers to disclose hazardous event chains and design countermeasures (e.g. interventions) in response to and for the prevention of accidents [Lev95]. *Fault tree analysis* (FTA) and *failure mode and effects analysis* (FMEA) specialise in the identification, assessment, and reduction of faults or failures. There are many variants and combinations of these techniques, enriched with intuitive models and visual languages (e.g. AcciMaps [SR02]; STAMP [Lev04] for systemtheoretic process analysis [Lev12]; UML for HazOp [GMGP10]; CORAS diagrams [LSS11]), tailored for specific stages in the system life cycle [Eri15], and used to shape assurance arguments [McD94, GC17].

Using FTA, one can produce a *fault tree* (FT), a versatile causal model of fault propagation in a system, relating an *undesired event* e_u with a set B of *basic events* connected by various *gates* (e.g. `AND`, `OR`, `NOT`). A *minimum cut set* (MCS) is a minimal set of events causing e_u. *Dynamic FTs* are an extension modelling the ordering of such events, leading to *minimal cut sequences*. Such sequences describe minimal sets of events to occur in a particular order to activate e_u. FTs can be expressed by connecting sets $minCS \subseteq 2^B$ in the disjunctive normal form in the antecedent and with e_u in the consequent of the implication $(\bigvee_{S \in minCS} \bigwedge_{e_b \in S} e_b) \Rightarrow e_u$, where *minCS* denotes all MCSs for e_u, and e_b stands for a *basic event*.

For *quantitative analysis*, FTA and FMEA have been further formalised using probabilistic models [Kum07, ORS06], particularly, input/output Markov chains [BCS07] for checking probabilistic temporal properties, or Markov automata [VJK16] for efficient calculation of failure rates. FTs can be synthesised from failure annotations of stochastic Petri nets [UPM18] or from counterexamples generated by continuous stochastic logic model checking of continuous time Markov chains [LFL13].

Risk handling. Many operational risks cannot be avoided or mitigated by passive measures (e.g. physical segregation, safe materials). Mechanical or electronic *equipment failures* are usually addressed by fault-tolerant, robust, or diverse designs [Bir17], *human errors* by ergonomic interfaces, controllers robust to unsafe inputs [Lev12], and *adverse environmental events* by bespoke safety functions (e.g. safety modes [AZI+17]). An elegant transition from formal FTs into the specification of such functions is described in [HRS98].

Run-time verification (RV) checks event traces recorded during operation for violation (safety) or acceptance (co-safety) of a property [LS09]. RV can range from simple watchdogs, the checking of constraints on complex algorithm outputs [BBH+17], to Bayesian prediction to warn about violated health parameters such as low battery time [IMM18] and violated probability thresholds for rear-end collisions [CLC+19]. Huang et al. [HEZ+14] show how *monitoring-oriented programming* (MOP) [MJG+11] allows the checking of safety properties in message communications of the *robot operating system*[7] (ROS). While these works focus on the synthesis of *property monitors*, there are also approaches to the design of *property enforcers* (e.g. [GPBB08]). Already in the 1980s, Sobek and Chatila [SC88] proposed the interruption of robot plan execution by *safety monitors* (e.g. for obstacle detection) responding with *corrective actions* (e.g. obstacle avoidance) and *mitigation monitors* resuming the planner after action success. Simmons [Sim94] speaks of deliberative components for normal situations and reactive behaviours for exceptional situations. Sorin et al. [SLJS16] describe the generation of safety monitors for ROS-based autonomous robots using rules with corrective actions.

Risk-aware control offers a versatile approach to property enforcement, for example, to minimise collision risk of *autonomous vehicles* (AVs). Althoff et al. [AKWB11] work with a probabilistic variant of inevitable collision states [FA04] to approximate collision probability and cost beyond the planning horizon by Monte Carlo sampling of trajectories. These metrics allow the ranking of simulated trajectories in navigation decisions. Pereira et al. [PBHS13] experiment with a grid-based minimal collision risk planner and a risk-aware *Markov decision process* (MDP) for navigating an underwater vehicle. Sanger [San14] formalises risk-aware movement by a risk estimate based on state uncertainty and the cost of control errors, implemented in a neuronal network-based AV controller. Feyzabadi and Carpin [FC14] propose a risk-aware planning algorithm using constrained MDPs, illustrated for autonomous indoor navigation. Müller and Sukhatme [MS14] encode collision risk by a Gamma distribution of the state and uncertain distance to a nearest obstacle. Shalev-Shwartz et al. [SSSS18] investigate collision-free navigation based on driving rules following the Duty of Care approach from Tort law [Har00]. This approach assumes proper response of other traffic participants in typical driving scenarios. The vehicle action space is discretised to simplify the control problem. Inspired by the notion and versatility of FTs, the present work seeks to identify the commonalities of works in risk-aware control and property enforcement and provide a unified risk handling framework.

[7] See https://www.ros.org

2.2. Formalism and notation

Our framework uses the *communicating sequential processes* (CSP, [Hoa85, Ros10]) approach to algebraic specification and *labelled transition system* (LTSs, [BK08]) to reason about its operational semantics. Let Σ be the set of all *concrete events* of a process, called its *alphabet*. We distinguish two special events: $\checkmark$ for the *termination* of a process and τ for the *invisible event*, resulting from *hiding* observations. We require $\checkmark, \tau \notin \Sigma$ and define $\Sigma^{\checkmark} = \Sigma \cup \{\checkmark\}$, $\Sigma^{\tau} = \Sigma \cup \{\tau\}$, and $\Sigma^{\checkmark,\tau} = \Sigma^{\checkmark} \cup \Sigma^{\tau}$.

Definition 1 (Process) A *process* is an expression of the form

$$P ::= a \rightarrow P \mid ?x : A \rightarrow P \mid P \sqcap P \mid P \,\Box\, P \mid P \underset{A}{\parallel} P \mid P; \; P \mid P \setminus A \mid SKIP \mid STOP \tag{1}$$

with event $a \in \Sigma$, shared alphabet $A \subseteq \Sigma$, event prefix ($\rightarrow$), prefix choice ($?x : \rightarrow$), non-deterministic choice ($\sqcap$), external choice ($\Box$), generalised parallel composition ($\parallel$), sequential composition ($;$), hiding ($\setminus$), termination ($SKIP$), and deadlock ($STOP$). Let $\mathcal{P}$ be the set of all *processes*.

Each expression $P \in \mathcal{P}$ corresponds to observable behaviours represented in a *trace model*. Traces are finite sequences over $\Sigma^{\checkmark}$ abstracting from details (e.g. internal states) of P's executions. In the trace model, $traces(P)$ yields the *trace semantics* of P. For example, $traces(STOP) = \{\langle\rangle\}$ and $traces(SKIP) = \{\langle\rangle, \langle\checkmark\rangle\}$. $initials(P) = \{a \mid \langle a\rangle \in traces(P)\}$ denotes the set of all *initial* events of P. For processes P and Q, we write $P \sqsubseteq_s Q$ to say that Q *s-refines* P (or P *is s-refined by* Q) and that $traces(P) \supseteq traces(Q)$, with the usual abbreviation $P =_s Q \iff P \sqsubseteq_s Q \wedge Q \sqsubseteq_s P$.[8] We use the term *control state* for a process and write $\parallel$ if $A = \Sigma$. Below, we consider $P = Sy \parallel En$, with a system Sy interacting with its environment En, or $P = En(Sy)$, with the CSP expression En using Sy.

Definition 2 (Labelled transition system) A LTS is a tuple $T = (S, \Sigma^T, \rightarrow, S_0)$ with a set S of *states*, a set $\Sigma^T \subseteq 2^{\Sigma^{\tau}} \setminus \{\varnothing\}$ of *abstract events* (i.e., sets of concrete events), a relation $\rightarrow \subseteq S \times \Sigma^T \times S$ of state transitions when engaging in these events, and a set $S_0 \subseteq S$ of *initial states*. Omission of S_0 results in an *uninitialised* LTS $(S, \Sigma^T, \rightarrow)$, further omission of Σ^T in a directed graph $(S, \rightarrow)$ with $\rightarrow \subseteq S \times S$, called *transition system* (TS). Let $\mathcal{T}$ be the set of all TSs.

We write $s \xrightarrow{e} s'$ if $(s, e, s') \in \rightarrow$, $s \xrightarrow{e}$ if $\exists\, s' \in S : (s, e, s') \in \rightarrow$, $s \rightarrow$ if $\exists\, e \in \Sigma^T, s' \in S : (s, e, s') \in \rightarrow$, $s \xrightarrow{e}\!\!\!\!\!\!\diagup$ if $\nexists\, s' \in S : (s, e, s') \in \rightarrow$, and $s \not\rightarrow$ if $\nexists\, e \in \Sigma^T, s' \in S : (s, e, s') \in \rightarrow$. A $\boxslash$ indicates discharged cases in proofs.

3. RISKSTRUCTURES: overview

In this section, we give an overview of the components of the proposed framework. We describe the abstraction facilitated by RISKSTRUCTURES, building a bridge between (probabilistic) process modelling, risk analysis, and controller design. Moreover, we illustrate the main concepts, terminology, and work steps using an example from the road vehicle domain, which will be revisited in later sections.

Figure 1 (bottom right) shows an example of a process P (Definition 1) associated with an LTS T (Definition 2). P describes actions of one or several agents (e.g. a machine, an operator) in a physical environment, for example, a manually driven vehicle with some kind of driving assistance on an urban road. P's actions include data processing, control stimuli via actuators, and the monitoring of process outcomes. In this example, data processing occurs with nav, where the machine performs calculations for navigation (stored internally and displayed to the driver), $step$ of the driver interpreting data for a next control action, $warn$ of the vehicle sending information to the driver, and $resume$ of the vehicle returning from a risk handling procedure. Control stimuli include the action drv, where the driver performs a default driving manoeuvre, $swBr$ where the vehicle executes a swerve and brake manoeuvre, and $emBr$ representing an emergency braking action conducted by the vehicle. Monitored outcomes include, for example, the occurrence ($e^{\mathsf{loc_d}}$) of the risk factor *loss of driving control* ($\mathsf{loc_d}$), a potential *accident* ($\underline{e}^{\mathsf{loc_d}}$) following $e^{\mathsf{loc_d}}$, and the performance and success of a mitigation ($m^{\mathsf{loc_d}}$) of this factor, recognised by the machine, and idling or no output (τ_i). Underspecification and uncertainty in P can be described by non-determinism or quantified by probabilities (λ_i) on action outcomes. $traces(P)$ denotes possible chains of events and T the reachable states (s_i).[9]

[8] We use $\sqsubseteq_T$ for *trace* refinement and $\sqsubseteq_{FD}$ for *failures-divergences* refinement and refer the inclined reader to [Ros10].

[9] $traces(P)$ contains finite traces reaching such states such that causes can be represented by well-founded sets.

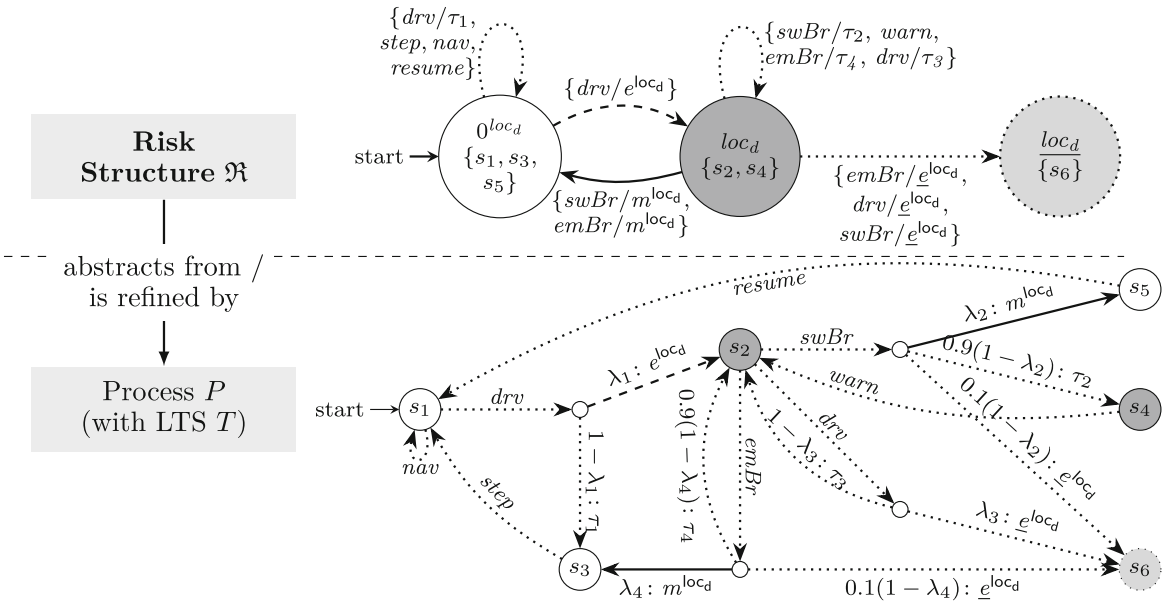


Fig. 1. Risk structure $\mathfrak{R}$ partitioning the states of the process P, that is, labelling P's states with the phases *inactive* (0^{loc}, white) and *active* (loc, dark gray) of the risk factor $\mathsf{loc_d}$; P's transitions are classified into endangerments (dashed arcs), mitigations (solid arcs), and other transitions (dotted arcs); incomplete abstractions may contain unclassified states (light-grey)

Risk, as introduced in Sect. 2.1, can then be qualitatively (or quantitatively) characterised in P as the possibility (probability) of an undesired state (e.g. s_4) being finitely reachable from a particular state (e.g. s_1) in T. Next, Fig. 1 shows the abstraction from P into an exemplary *risk structure* $\mathfrak{R}$ comprising the single factor $\mathsf{loc_d}$. This abstraction allows one to focus on the events relevant from the perspective of this factor, that is, its activation ($e^{\mathsf{loc_d}}$) and mitigation ($m^{\mathsf{loc_d}}$). Accordingly, this abstraction step includes the labelling of regions of P's state space with predicates (e.g. for hazards and, more generally, situations) and the construction of $\mathfrak{R}$ over the resulting *abstract state space*. Moreover, this representation helps one to focus on the structure of risk of a certain application by using information about factors encoded in predicates (e.g. active, inactive) and real-valued functions (e.g. [AKWB11, PBHS13]). These predicates and functions can be derived from hazard analysis (Sect. 2.1) and be used to allow a risk-aware machine to estimate risk levels of reachable states and to plan and execute actions reaching or avoiding the corresponding states.

For example, the factor $\mathsf{loc_d}$ partitions T's state space (cf. nodes in the bottom right of Fig. 1) into regions where $\mathsf{loc_d}$ is *inactive* (0^{loc_d}, white), *active* (loc_d, dark grey), or where an *accident* ($\underline{loc_d}$, light-grey) has happened. To focus on the practical handling of $\mathsf{loc_d}$ in P, outcomes (e.g. $e^{\mathsf{loc_d}}$, $m^{\mathsf{loc_d}}$, τ_i) are paired with their control actions (e.g. drv, $swBr$, $emBr$; dotted arcs) and merged into sets of transition labels called events (e.g. $\{swBr/m^{\mathsf{loc_d}}, \dots\}$, $\{swBr/\tau_2, \dots\}$). To keep track of mishap states, such as s_6, we assign the label $\underline{loc_d}$ to states where undesired consequences after an unmitigated $e^{\mathsf{loc_d}}$-event will be materialised. Such states might be reached, for example, after the execution of $emBr$, drv, and $swBr$ with a *collision* $\underline{e}^{\mathsf{loc_d}}$ as the possible outcome. $\underline{loc_d}$ enables risk assessment in terms of likelihood and severity. To design mitigations for $\underline{loc_d}$, we introduce an extra risk factor (e.g. col) handling $\underline{e}^{\mathsf{lod_d}}$ in analogy to $e^{\mathsf{loc_d}}$. This technique will be discussed below. Overall, for $\mathsf{loc_d}$, $\mathfrak{R}$ reduces to just two abstract states (0^{loc_d}, loc_d) and two events, the endangerment $e^{\mathsf{loc_d}}$ (dashed arc) and the mitigation $m^{\mathsf{loc_d}}$ (solid arcs). We assume there to be observational *refinement* or some simulation relation between T and $\mathfrak{R}$.

Overall, risk awareness amounts to the machine incorporating $\mathfrak{R}$ in its decisions such that expected risk from its own actions does not exceed an acceptable level. *So, how can we engineer* $\mathfrak{R}$*?*

From an engineering perspective, our approach requires neither P nor $\mathfrak{R}$ to exist in a complete form to facilitate the aforementioned abstraction or refinement. One can rather think of these two artefacts as being developed concurrently.

Table 1. A generic situation/action/factor table (a) and per-cell risk analysis checklists (b, c)

	Action	a_1		a_2		...
Situation	**Factor**	$\mathsf{f}_{1.1}$	$\mathsf{f}_{1.2}$ ...	$\mathsf{f}_{2.1}$	$\mathsf{f}_{2.2}$ ...	...
(a) Generic situation/action/factor table						
σ_1		Each cell filled with results from a cause/consequence and				...
σ_2		mitigation analysis along the lines of Tables 1b and 1c.				...
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

(b) Factor activation analysis (causes/consequences)

Why would f_i occur with/during action a_j in situation σ_k?
What if f_i occurs with/during action a_j in situation σ_k?

1. What is the likelihood of occurrence of f_i?
Quantified by $Pr[f_i \mid a_j, \sigma_k]$.
2. What is the risk of an accident $\underline{f_i}$ following f_i?
Quantified by the likelihood $Pr[\underline{f_i} \mid f_i, a_j, \sigma_k]$ and
the severity $sev(\underline{f_i};\ f_i, a_j, \sigma_k)$.
3. Which other factors are related to the activation of f_i?
Qualified by relationships such as f_i `requires` f_j, f_i `causes` f_k; and quantified by $Pr[f_k \mid f_i, a_j, \sigma_k]$.

(c) Factor mitigation analysis

How can we mitigate f_i during or after action a_j in the situation σ_k?

4. Which mitigation options m are available to reach $\overline{f}_i$?
5. What is the risk of an accident $\underline{f_i}$ following $\overline{f}_i$?
Quantified by the likelihood $Pr[\underline{f_i} \mid \overline{f}_i, m, a_j, \sigma_k]$ and
the severity $sev(\underline{f_i};\ \overline{f}_i, m, a_j, \sigma_k)$.
6. Which other factors are related to the mitigation of f_i?
Qualified by relationships such as f_i `prevents` f_j, f_i `causes` f_k; and quantified by $Pr[f_k \mid \overline{f}_i, m, a_j, \sigma_k]$.

For example, one might start with a part of P, then, after risk analysis, continue with a part of $\mathfrak{R}$, and in a second iteration, maybe even after a period of operation, proceed with another part of both P and $\mathfrak{R}$. In this sense, $\mathfrak{R}$ can be incrementally developed. The notion of a situation will help us to structure such a development. A *situation*, σ, is an abstract state that describes a particular context (i.e., a scene or environment), an activity (i.e., a task, operating mode, a use case, or scenario), and a configuration of hazardous events occurring in P. Engineers and risk analysts are familiar with tables and graphs, for example, when applying HazOp or FMEA to assess risk factors of certain machine actions in particular situations. We will, therefore, show in the following how one can develop $\mathfrak{R}$ by means of what we call a *situation/action/factor* table (Table 1a).

One may want to start with an initial set of application-specific *situations* and *actions* that the machine under consideration can perform, together with an initial set of *factors*, maybe obvious from accident experience and preliminary hazard analyses. Then, each resulting cell in Table 1a would involve a focused cause/consequence analysis (e.g. [ORS06]) and risk assessment (Sect. 2.1), for example, using risk graphs [Kum07, p. 53]. Tables 1b and 1c elaborate ideas in [Gle18] for such analyses, including the estimation of the conditional probability $Pr[X \mid Y]$ and severity $sev(X;\ Y)$ of an event X under the condition Y. We use 0^f, f, $\overline{f}$, and $\underline{f}$ to denote the *inactive*, *active*, *mitigated*, and *mishap* phases of a factor f, explained in Sect. 4.1. Incrementally, one would add further situations, actions, and factors to the table. At some point, one can split the table into several smaller ones. Please, note that such tables are not subject of this work. We presented them merely to hint to the data to be collected for building $\mathfrak{R}$. Moreover, when formalising situations, we will only consider the hazardous events part. In our future work, we will discuss situations in more detail.

Table 2. Example of a situation/action/factor table for road driving

Situation	Action / Factor / Descr.	drive (drv) $\mathsf{loc_d}$ loss of control	ncol near-collision	col collision	releaseAirbag (rA) $\mathsf{st_{rA}}$ spurious trip	$\mathsf{fod_{rA}}$ fail. on dem.
`manual` mode		3. Pr[ncol | $\mathsf{loc_d}$]: h, `requires` $\mathsf{st_{rA}}$	3. Pr[col | ncol]: h 4. swerve; brake	2. $sev(\underline{col})$: m 3. `requires` ncol	2. $sev(shock)$: m 3. `causes` $\mathsf{loc_d}$	–
`autonomous` mode		–	3. Pr[col | ncol]: h 4. swerve; brake	2. $sev(\underline{col})$: m 3. `requires` ncol	2. $sev(shock)$: m	–
`following` vehicle		3. equal to `manual/autonomous`		2. $sev(\underline{col})$: l	3. equal to `manual/autonomous`	
`leading` vehicle		3. equal to `manual/autonomous`		2. $sev(\underline{col})$: h, 4. rA, 5. $sev(\underline{col})$: m	3. equal to `manual/autonomous`	
`oncoming` vehicle		3. equal to `manual/autonomous`		2. $sev(\underline{col})$: h, 4. rA, 5. $sev(\underline{col})$: h	3. equal to `manual/autonomous`	
`collision` (col)		–	–	2. $sev(\underline{col})$: m	–	3. equal to *
`no airbag` ($\mathsf{fod_{rA}}$)		1. Pr[$\mathsf{loc_d}$]: l 3. eq. to `manual`	3. eq. to `manual`	* 2. $sev(\underline{col})$: h	–	3. idempotent

Legend: Numbers indicate the analysis steps applied from the Table 1b and 1c; situation/action/factor parameters for Pr and sev are determined by the cell (hence omitted); values after the colon are exemplary; probabilities are abstracted to $h/m/l$ for high/medium/low; relationship sources are determined by the column (hence omitted); *. . . cross-reference to other cell;–irrelevant/not applicable

Extending the Example from Fig. 1 In Fig. 1, we illustrated basic concepts by discussing a single risk factor ($\mathsf{loc_d}$). Now, we extend our discussion to a more realistic setting by investigating situations, system actions, and further risk factors. In Table 2, we consider actions such as `releaseAirbag` (rA), responsible for mitigating a collision col, that is, $m^{\mathsf{col}} \equiv rA$. However, rA itself may be associated with two failure modes: *failure on demand* $\mathsf{fod_{rA}}$ (i.e., rA not performed when requested) and *spurious trip* $\mathsf{st_{rA}}$ (i.e., rA performed when not requested). In forward reasoning, one can ask: What if $\mathsf{fod_{rA}}$ or $\mathsf{st_{rA}}$, both influencing or determining the behaviour of rA, are activated in `manual` or `autonomous` driving mode? $\mathsf{st_{rA}}$ in `manual` mode would increase the probability of a crash because of a *loss of control* $\mathsf{loc_d}$ from shock and distraction. $\mathsf{st_{rA}}$ in `autonomous` mode may cause shock injuries but would unlikely cause a crash. $\mathsf{fod_{rA}}$ will be irrelevant in normal situations but in a `collision`, risk will be quite similar to the case of `no airbag`.

Table 2 can store probabilistic relationships, for example, between *near-collision* ncol and *collision* col. Probabilities for ncol and col will be highly dynamic and need to be estimated and used for decision making at run-time. Probabilities for $\mathsf{fod_{rA}}$ maybe more stable, known from experiments, and can be used in risk assessment at design-time. Requirements on the confidence in probability estimates are specific to each factor. Note the variety of relationships between situations, actions, and factors, for example, in `manual` mode $\mathsf{loc_d}$ `requires` $\mathsf{st_{rA}}$, $\mathsf{fod_{rA}}$ in `collision` *is similar to* col in `no airbag`, $\mathsf{st_{rA}}$ in `manual` mode *extends* $\mathsf{st_{rA}}$ in `autonomous` mode, and the airbag, subject of risk assessment in the two rA-columns, mitigates col.

Graphs (e.g. FTs) can be used to visualise and elicit the information stored in Table 2 if these tables get more complex. The tabular representation is, however, integrative, scales, and conveys the scheme recurring across the cells as well as the relationships between situations, actions, and factors. We will discuss and use some of these relationships in our framework.

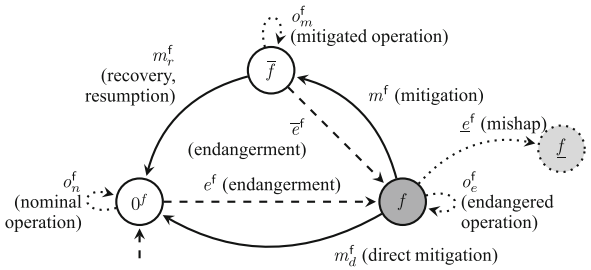
The components of a design algebra for risk-aware machines will be described in the following sections: risk factors and risk spaces (Sect. 4), mitigation orders (Sect. 5), factor dependencies (Sect. 6), and, finally, RISKSTRUCTURES (Sect. 7), illustrated by examples and a discussion of applications.

4. Risk spaces

This section introduces *risk factors* and *risk spaces*, frequently abbreviated by "factors" and "spaces", as the basic elements of the algebraic framework.

4.1. Risk factors

We first define the notion of a risk factor using a LTS and describe its properties and meaning.



(a) LTS for a risk factor f

(b) Events abstracting from P to the factor f

Event	Concrete events in P ...
e^{f}	from nominal to endangered operation
$\overline{e}^{\mathsf{f}}$	from mitigated to endangered operation
$\underline{e}^{\mathsf{f}}$	from endangered operation to mishap
m^{f}	from endangered to mitigated operation
m^{f}_d	from endangered to nominal operation
m^{f}_r	from mitigated to nominal operation
o^{f}_n	preserving 0^f
o^{f}_e	preserving f
o^{f}_m	preserving $\overline{f}$

Fig. 2. Basic template of a risk factor f (a) with a description of f's events (b)

Definition 3 (Risk factor) Let $(Ph_{\mathsf{f}}, \Sigma^{\mathsf{f}}, \rightarrow_{\mathsf{f}})$ be an uninitialised LTS according to Definition 2. Extending this LTS, a *risk factor* is a tuple $\mathsf{f} = (Ph_{\mathsf{f}}, \Sigma^{\mathsf{f}}, \rightarrow_{\mathsf{f}}, \preceq_{\mathsf{f}}, s_{\mathsf{f}})$ with

- A finite set Ph_{f} of *phases* of f,
- A finite set $\Sigma^{\mathsf{f}} \subset 2^{\Sigma^{\tau}} \setminus \{\varnothing\}$ specifying *significant events* for f,
- A labelled transition relation $\rightarrow \subseteq Ph_{\mathsf{f}} \times \Sigma^{\mathsf{f}} \times Ph_{\mathsf{f}}$,
- A partial order $\preceq_{\mathsf{f}} \preceq_{\mathsf{f}} \subseteq Ph_{\mathsf{f}} \times Ph_{\mathsf{f}}$ called *phase order* of f, and
- A pair $s_{\mathsf{f}} \in \mathbb{R}^2_+$ with $s^{(1)}_{\mathsf{f}} \leq s^{(2)}_{\mathsf{f}}$ denoting the *severity of the least and worst expected impact* of f.[10]

Let $\mathcal{F}$ be the set of all risk factors.

We consider factors f with $\rightarrow_{\mathsf{f}}$ according to Fig. 2a with $Ph_{\mathsf{f}} = \{0^f, f, \overline{f}\}$ for the phases *inactive* (0^f, typically, the initial phase), *active* (f), and *mitigated* ($\overline{f}$), where $\preceq_{\mathsf{f}} \preceq_{\mathsf{f}}$ is at least[11] the reflexive transitive closure of $\{(f, 0^f), (f, \overline{f})\}$, and with $\Sigma^{\mathsf{f}} = \{e^{\mathsf{f}}, \overline{e}^{\mathsf{f}}, m^{\mathsf{f}}, m^{\mathsf{f}}_d, m^{\mathsf{f}}_r, o^{\mathsf{f}}_n, o^{\mathsf{f}}_m, o^{\mathsf{f}}_e\}$.

The interval s_{f} represents the *severity* (also known as the detriment or negativity) of f's expected materialised impact or consequences for any relevant assets if f gets and stays active. Impact can include, for example, damage of the environment, injury of humans, loss of a valuable, or any combination of these.

Events and factor types. By Definition 3, for any $p, p' \in Ph_{\mathsf{f}}$ if $(p, e, p') \in \rightarrow_{\mathsf{f}}$ then $e \neq \varnothing$. f is *deterministic* if and only if for every phase the events of all outgoing transitions are pairwise disjoint. We require $\forall p \in Ph_{\mathsf{f}} \colon \left(\bigcup_{(p,e,p') \in p \rightarrow} e\right) \setminus \{\tau\} = \Sigma$.[12] Figures 2a and 2b indicate the meaning of the events in Σ^{f} for factor modelling. The three phase-preserving events o^{f}_n, o^{f}_e, and o^{f}_m complement the endangerment and mitigation events. Moreover, we distinguish some special types of factors:

- If $m^{\mathsf{f}} \cup m^{\mathsf{f}}_d = \varnothing$ then we call f *final*, otherwise *reducible*.
- For any reducible f with $m^{\mathsf{f}} \neq \varnothing$, if $\overline{e}^{\mathsf{f}} \subset e^{\mathsf{f}}$ then we call f *strongly reducible*.
- If $m^{\mathsf{f}}_d = \varnothing$ then we call f *indirectly reducible*.

Modelling mishaps. Observing from the example in Sect. 3, the *mishap* phase $\underline{f}$, modelling incidents or accidents from f, complements the severity information encoded in $\mathsf{f}.s$ and enables risk assessment solely based on a single risk factor. It turns out, however, that $\underline{f}$ is non-essential for the basic framework we want to discuss in the following sections. While we keep $\underline{f}$ for factor-specific risk assessment, to simplify algebraic reasoning about factors as discussed below, we collapse $\underline{f}$ into f and $\underline{e}^{\mathsf{f}}$ into o^{f}_e. Furthermore, we introduce a final factor f' to model the corresponding mishap. This approach simplifies the basic factor model and allows the conversion of f' into a reducible factor when a mitigation for f' is available. For example, as illustrated in Table 2, the factor col provides accident handling for the factor ncol, whose likelihood of occurrence under the condition of $\mathsf{loc_d}$ is high. In [Gle18, GC20], however, we have started to work with a refined factor model including $\underline{f}$ as well

[10] By usual convention, for an ordered n-tuple t and $i \in [1..n]$, we write $t^{(i)}$ to refer to the value of the i-th element of t. Furthermore, if t has a uniquely named element e, we write $t.e$ to refer to the value of e in t.

[11] Some applications might give rise to a *linear* $\preceq_{\mathsf{f}} \preceq_{\mathsf{f}}$ by adding at most one out of $\{(0^f, \overline{f}), (\overline{f}, 0^f)\}$.

[12] In testing, this is known as input-enabledness [Tre08].

as further phases and events (e.g. for mishap alleviation). In order to preserve the main results presented here, such extensions require $\preceq_{\mathsf{f}} \preceq_{\mathsf{f}}$ to have a unique maximal element (discussed in Sect. 5.3) and an adjustment of constraint definitions in Sect. 6.1. We conclude that Ph_{f} and Σ^{f} comprise a useful minimal set of elements for a *generic* risk factor f.

4.2. Risk states, spaces, and space composition

Risk factors give rise to *risk states* and *risk spaces*. Let $F \subset \mathcal{F}$ be a finite factor set (Definition 3) where each $\mathsf{f} \in F$ has the form $\mathsf{f} = (Ph_{\mathsf{f}}, \Sigma^{\mathsf{f}}, \rightarrow_{\mathsf{f}}, \preceq_{\mathsf{f}} \preceq_{\mathsf{f}}, s_{\mathsf{f}})$.

Definition 4 (Risk state) Assume that risk factors are unique, that is, $\forall\, \mathsf{f}, \mathsf{g} \in F : \mathsf{f} \neq \mathsf{g} \Rightarrow Ph_{\mathsf{f}} \cap Ph_{\mathsf{g}} = \varnothing$. Then, a *risk state* is a faithful total injection $\sigma : F \to \bigcup_{\mathsf{f} \in F} Ph_{\mathsf{f}}$, that is, $\forall\, \mathsf{f} \in F : \sigma(\mathsf{f}) \in Ph_{\mathsf{f}}$.

A risk state abstracts from states of P by focusing on risk-related information in form of state propositions associated with the factor phases. Such propositions will, however, not be formalised in this work.

Definition 5 (Risk space) For a factor set F, a *risk space* $R(F)$ is the function space given by

$$R(F) = \{\sigma \in F \to \bigcup_{\mathsf{f} \in F} Ph_{\mathsf{f}} \mid \sigma \text{ is total} \wedge \sigma \text{ is an injection} \wedge \forall\, \mathsf{f} \in F : \sigma(\mathsf{f}) \in Ph_{\mathsf{f}}\}\ .$$

We omit the parameter F from R if it is clear from the context and denote the set of all risk spaces by $\mathcal{R}$.

By Definition 5, R is non-empty and finite if and only if F is non-empty and finite. R defines the set of all states an arbitrary[13] combination of factor phases might give rise to. Note that $\sigma(\mathsf{f})$ can now be used to refer to the phase of factor f in a risk state σ.

Definition 6 (Compatibility of risk states) Given $F_1, F_2 \subset \mathcal{F}$, $\sigma \in R(F_1)$ and $\sigma' \in R(F_2)$ are *compatible*, written

$$\sigma \approx \sigma' \iff \forall\, \mathsf{f} \in F_1 \cap F_2 : \sigma(\mathsf{f}) = \sigma'(\mathsf{f})\ . \tag{2}$$

Notice that any two risk states $\sigma \in R(F_1)$ and $\sigma' \in R(F_2)$ are compatible if $F_1 \cap F_2 = \varnothing$ and, furthermore, that state equality implies $F_1 = F_2$ and, therefore, state compatibility. This compatibility is a prerequisite for risk space composition as follows.

Definition 7 (Risk space composition) The *composition* $\otimes : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ of the two risk spaces $R(F_1)$ and $R(F_2)$ is defined by

$$R(F_1) \otimes R(F_2) = \{\sigma_1 \cup \sigma_2 \mid \sigma_1 \in R(F_1) \wedge \sigma_2 \in R(F_2) \wedge \sigma_1 \approx \sigma_2\}. \tag{3}$$

Now, we can derive a basic law relating the union of risk factors and the composition of risk spaces. Furthermore, it will turn out that R is a homomorphism.

Lemma 1 (Exchange of $\cup$ and $\otimes$)

$$R(F_1 \cup F_2) = R(F_1) \otimes R(F_2)$$

Proof sketch. The proof is by mutual existence and uniqueness: For each $\sigma \in R(F_1 \cup F_2)$, (i) there exists a $\sigma_1 \cup \sigma_2 \in R(F_1) \otimes R(F_2)$ and (ii) this pair is unique, and (iii, iv) conversely. Details on the proof can be taken from Appendix A. □

Lemma 2 (R is homomorphic) R is a homomorphism in the context of $(\mathcal{F}, \cup)$ and $(\mathcal{R}, \otimes)$.

$$\begin{array}{ccc} F_1, F_2 & \xrightarrow{\cup} & F_1 \cup F_2 \\ \downarrow R & & \downarrow R \\ R(F_1), R(F_2) & \xrightarrow{\otimes} & R(F_1 \cup F_2) \end{array}$$

[13] Below, we also view R as "the most general (risk) structure" for a specific factor set.

Proof sketch. We first make sure that we deal with semi-groups and then show by algebraic manipulation that $\otimes$ is associative. Details on the proof can be taken from Appendix A. □

Two specific classes of risk states and a basic lemma close this section. $R(\{\mathsf{f}\}) = \{[\mathsf{f} \mapsto 0^f], [\mathsf{f} \mapsto f], [\mathsf{f} \mapsto \overline{f}]\}$ forms the *trivial risk space* for f, and $R(\varnothing) = \varnothing$ the *empty risk space*, used in the following equality.

Corollary 1 For finite $F \subset \mathcal{F}$, Lemma 1 yields $R(\varnothing)$ to be the *zero* element of composition with $\otimes$:

$$R(\varnothing) \otimes R(F) = R(F)\,. \qquad (\otimes\text{-zero})$$

4.3. Example: risk factors on the road

In the following, we discuss Table 2 more deeply and provide further examples for the event sets in Fig. 2b. Tracing back through the causal relation from col, we find that *collisions* require *near-collisions* to occur beforehand. Based on that observation, we identify a combination of *swerve & brake* ($swBr$) to be a possible mitigation for ncol. However, the action $swBr$ gives rise to the *failure on demand* $\mathsf{fod_{br}}$ of the car's braking action br. This extension of the analysis in Table 2 results in another dependency, namely, $\mathsf{fod_{br}}$ *impedes the mitigation of* ncol, technically, it disables the braking action br.

Tracing forward through the causal relation, we may stop at application-specific factors. Such factors can be modelled as *final*, that is, the machine will not mitigate them. This way, final factors define the *scope* of $\mathfrak{R}$. However, as explained in Sect. 4.1, a design increment of the machine can turn a final into a *reducible* factor. For example, designing an airbag into a car makes col reducible. This epistemic limit explains why safety always has to be considered *relative* to a known factor set F [Gle14].

Tracing back again from $\mathsf{fod_{br}}$ discloses the failure mode *degradation of brake* $\mathsf{deg_{br}}$. $\mathsf{deg_{br}}$ causally relates to $\mathsf{fod_{br}}$ like ncol relates to col: $\mathsf{deg_{br}}$ will make br ineffectual but not ineffective, if one does not intervene, $\mathsf{deg_{br}}$ will actually cause $\mathsf{fod_{br}}$. We want $\mathsf{deg_{br}}$ to be *strongly reducible*, based on a mitigation $m^{\mathsf{deg_{br}}}$ that establishes $\overline{deg_{br}}$ from which only a strict subset $\overline{e}^{\mathsf{deg_{br}}} \subset e^{\mathsf{deg_{br}}}$ can occur. In phase deg_{br}, *drive by and halt* would be a conservative option for $m^{\mathsf{deg_{br}}}$ to be implemented by a safety controller. These reasoning steps are summarised in Fig. 3.

While the state bi-partition for $\mathsf{loc_d}$ shown in Fig. 1 seems sufficient, some factors need a tri- or n-partition. For example, $\mathsf{fod_{br}}$ suggests a vehicle repair. Therefore, Fig. 2a introduces a third phase to separate the state where a brake failure ($\mathsf{fod_{br}}$) is mitigated ($\overline{fod_{br}}$) from the state $0^{fod_{br}}$ where the brake is fully operational. Because $\mathsf{fod_{br}}$ requires an off-line repair action $m_r^{\mathsf{fod_{br}}}$, it is *indirectly reducible*. Two further examples for such a factor would be *leaking or damaged battery* and *run out of fuel*. Reaching the "safe state" $\overline{fod_{br}}$ is possible via an intermediate stable state such as *halted at car repair shop*. From this phase, recovery ($m_r^{\mathsf{fod_{br}}}$) to the inactive phase $0^{fod_{br}}$ should be feasible. In contrast, ncol or *too close to the front vehicle* are factors that can often be dealt with by braking or swerving correspondingly (i.e., $swBr$). Thus, reaching 0^{ncol} should be possible. In this sense, ncol is *directly reducible*.

Risk awareness emerges from combining several factors (e.g. $F = \{\mathsf{loc_d}, \mathsf{ncol}, \mathsf{col}, \mathsf{st_{rA}}, \mathsf{fod_{rA}}, \mathsf{fod_{br}}, \mathsf{deg_{br}}\}$) into a risk space $R(F)$, detecting the current risk state, and estimating the likelihood of neighbouring, potentially worse, risk states. Our example suggests that risk awareness contains a notion of machine health. One might expect an autonomous vehicle to take full responsibility of emergency control in any of the states in $R(F)$.

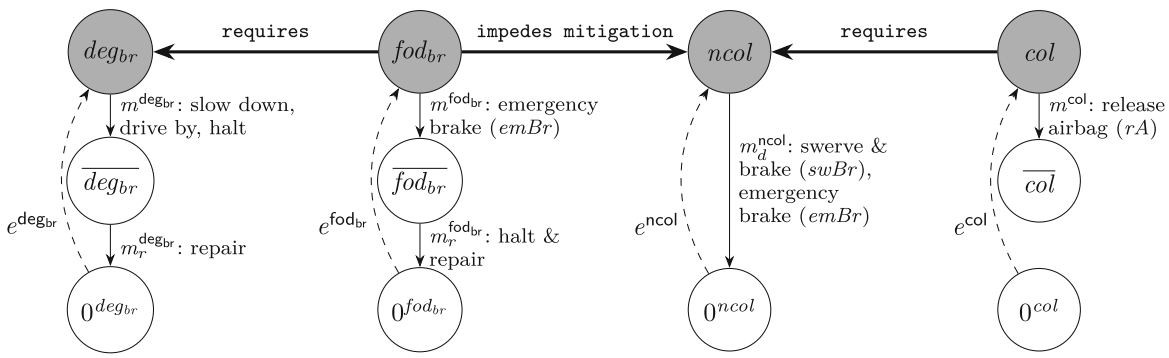


Fig. 3. Factor instances ($\mathsf{deg_{br}}$, $\mathsf{fod_{br}}$, ncol, col) and their dependencies (thick arcs, e.g. `requires`)

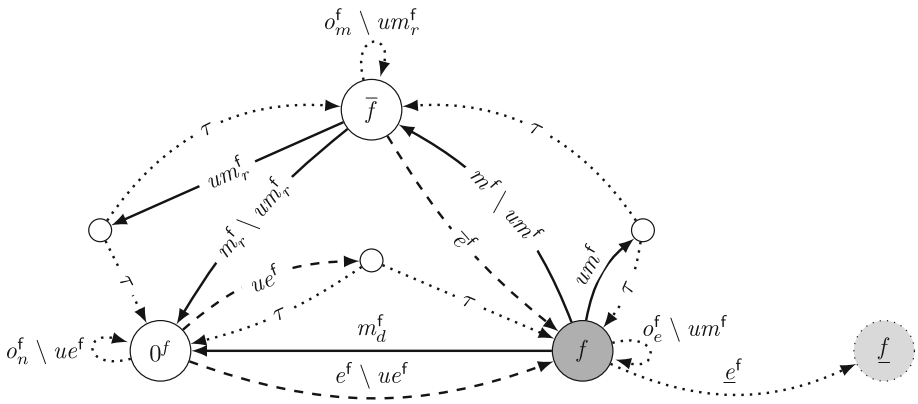


Fig. 4. Risk factor f with non-deterministic fractions ue^{f}, um^{f}, and um^{f}_r. The label τ is to be read as $\{\tau\}$

This example illustrates how the proposed formalism captures and guides the way of thinking of safety engineers responsible for developing and assuring safety controllers of autonomous machines.

4.4. Discussion: abstraction and types of risk factors

Types of risk factors and risk states Factors can be used to model faults, failure modes, hazards, incidents, and accidents. Final factors can model, for example, *permanent and off-line repairable faults*, and reducible factors, for example, *transient and on-line repairable faults*.

States with an *active final factor* f (Sect. 4.1) expose the process P to *residual risk infinitely long and often*, thus, almost certainly materialising the consequences of f. By using $\mathfrak{R}$, a risk-aware machine in a process P should govern its (often probabilistic) choices to not enter such states.

Uncertainty in risk factors Figure 1 illustrates the abstraction from a possibly probabilistic process P into a deterministic risk structure $\mathfrak{R}$ with a single factor, now call it f. This abstraction could preserve the probability of occurrence of f, if estimated, in $\mathfrak{R}$. However, the quantities of f (especially the probabilities) will often not be confidently known even if they are continuously estimated during operation. But, as indicated in Fig. 4, assumptions about the events of f might still be made, for example,

1. uncertainty about the actual occurrence of an endangerment e^{f} expressed as $ue^{\mathsf{f}} = o^{\mathsf{f}}_n \cap e^{\mathsf{f}}$,
2. uncertainties $um^{\mathsf{f}} = o^{\mathsf{f}}_e \cap m^{\mathsf{f}}$ and $um^{\mathsf{f}}_r = o^{\mathsf{f}}_m \cap m^{\mathsf{f}}_r$ in the success of mitigation m^{f} and recovery m^{f}_r, or
3. the likelihood of an accident $\underline{f}$ from f.

Deterministic risk factors ($ue^{\mathsf{f}} = \varnothing$, Definition 3) assume that $\mathfrak{R}$ is certain about the preconditions that activate f, that is, from observing e^{f}, $\mathfrak{R}$ knows that the machine enters f. However, $ue^{\mathsf{f}} \neq \varnothing$ means that $\mathfrak{R}$ can observe e^{f} but P does not necessarily move to a state fulfilling f.

An example for 1. could be an obstacle tracked within safe braking distance with low confidence, justifying both states 0^f and f. This situation requires further state estimation (e.g. a homing algorithm [NSV03]) to disclose further information about the tracked obstacle. We also discuss this below in Sect. 7.6.

Moreover, $e^{\mathsf{f}} \setminus ue^{\mathsf{f}}$ can describe known causes of f and ue^{f} *potential* causes of f. $o^{\mathsf{f}}_n \supseteq e^{\mathsf{f}}$ means that $\mathfrak{R}$ is uncertain about all of f's causes. Following [Ros10, p. 116], ue^{f} is followed by non-deterministic choice over 0^f and f, modelled by τ from an anonymous state (○) in the factor LTS. Analogously, with um^{f} for f and um^{f}_r for $\overline{f}$. The factor in Fig. 4 is equivalent to the one in Fig. 2a according to the following lemma.

Lemma 3 (Isolating uncertainty preserves factor properties)

$$\mathsf{f}_{(4)} =_{FD} \mathsf{f}_{(2a)}$$

Proof sketch. The proof is by showing that in both cases, considering uncertainty explicitly by isolating non-determinism (Fig. 4) and considering uncertainty implicitly by not isolating non-determinism (Fig. 2a), we deal with the same factor model. Details on the proof can be taken from Appendix A. □

For the sake of simplicity of the discussion, we neglect further non-determinism possible for the three other actions $\vec{e}^{\mathsf{f}}$, m_d^{f}, and $\underline{e}^{\mathsf{f}}$. However, our discussion suggests that in future work we can explore a factor theory addressing those factor types that are likely to occur frequently in practical applications.

5. Mitigation orders

This section investigates three basic orders over risk spaces, all intended to support the evaluation (e.g. comparison) of successors of a particular risk state reached after further endangerments or deployed mitigations. The *fully and partially comparable inclusive mitigation orders* (Sect. 5.1), both partial orders, are based on qualitative information, and the *strong mitigation order* (Sect. 5.2), a linear order, is based on quantitative information. These orders depend on the available information about risk and are related (Sect. 5.3).

5.1. Qualitative mitigation orders

Let $R(F)$ be a risk space (Definition 5) for a factor set $F \subseteq \mathcal{F}$. Then, we define a partial order $\preceq_{\mathsf{m}} \subseteq R \times R$ as follows.

Definition 8 (Fully comparable inclusive mitigation order) For any pair of states $\sigma, \sigma' \in R$, define

$$\sigma \preceq_{\mathsf{m}} \sigma' \iff \forall \mathsf{f} \in F\colon \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) \,.$$

By $\sigma \prec_{\mathsf{m}} \sigma' \iff \sigma \preceq_{\mathsf{m}} \sigma' \wedge \sigma \neq_{\mathsf{m}} \sigma'$, we induce the corresponding strict order. σ and σ' are said to be *incomparable* if and only if $\sigma \not\preceq_{\mathsf{m}} \sigma' \wedge \sigma' \not\preceq_{\mathsf{m}} \sigma$. Intuitively, $\sigma \preceq_{\mathsf{m}} \sigma'$ signifies that "σ' is a better achievement in risk mitigation than σ."[14] However, $\preceq_{\mathsf{m}}$ requires full comparability of two states. It might be cumbersome to require such comprehensive knowledge to determine which state is "more or less risky" than another. Allowing for both epistemic and aleatory uncertainty, we might instead account for partial knowledge in the phase orders (Definition 3) at the level of R by providing a relaxed partial order as follows.

Definition 9 (Partially comparable inclusive mitigation order) For states $\sigma, \sigma' \in R$, define

$$\sigma \precsim_{\mathsf{m}} \sigma' \iff \forall \mathsf{f} \in F\colon \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) \vee \left((\sigma(\mathsf{f}), \sigma'(\mathsf{f})) \notin \preceq_{\mathsf{f}} \wedge (\sigma'(\mathsf{f}), \sigma(\mathsf{f})) \notin \preceq_{\mathsf{f}}\right) .$$

We use $\prec_{\mathsf{m}}^{\sim}$ and $=_{\mathsf{m}}^{\sim}$ to distinguish the corresponding strict order and equality for $\precsim_{\mathsf{m}}$ from $\prec_{\mathsf{m}}$. Intuitively, Definition 9 requires a "betterment in risk from σ to σ'" based exactly on the comparable phases.

Lemma 4 For any pair of risk states $\sigma, \sigma' \in R$, we have that

$$\sigma \preceq_{\mathsf{m}} \sigma' \Rightarrow \sigma \precsim_{\mathsf{m}} \sigma' \,.$$

Proof of Lemma 4. $\preceq_{\mathsf{f}}$ is antisymmetric. By definition of $\preceq_{\mathsf{m}}$, we may assume

$$\begin{aligned}
&\forall \mathsf{f} \in F\colon \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) && (\forall\text{-elim})\\
&\vdash \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) && (\vee\text{-intro1})\\
&\vdash \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) \vee \left((\sigma(\mathsf{f}), \sigma'(\mathsf{f})) \notin \preceq_{\mathsf{f}} \wedge\ (\sigma'(\mathsf{f}), \sigma(\mathsf{f})) \notin \preceq_{\mathsf{f}}\right) && (\forall\text{-intro, assumption for each f})\\
&\vdash \forall \mathsf{f} \in F\colon \sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f}) \vee \left((\sigma(\mathsf{f}), \sigma'(\mathsf{f})) \notin \preceq_{\mathsf{f}} \wedge\ (\sigma'(\mathsf{f}), \sigma(\mathsf{f})) \notin \preceq_{\mathsf{f}}\right) && \square
\end{aligned}$$

Corollary 2

$$\sigma' \neq_{\mathsf{m}}^{\sim} \sigma \Rightarrow \sigma' \neq_{\mathsf{m}} \sigma$$

[14] This notation might feel unusual as risk reduction is about lowering risk. So, if σ' is a state with *lower risk* than state σ then we could write "$\sigma \succeq_{\mathsf{m}} \sigma'$". But we might agree that being in σ' is better than residing in σ. Moreover, for risk mitigation, reaching *better* states from worse is in the foreground. Thus, it seems reasonable to use the inverted notation $\sigma \preceq_{\mathsf{m}} \sigma'$.

Proof of Corollary 2.

$$\sigma' \neq^{\sim}_{\mathsf{m}} \sigma \Rightarrow \sigma' \neq_{\mathsf{m}} \sigma \qquad \text{(by definition)}$$
$$\neg(\sigma' \precsim_{\mathsf{m}} \sigma \wedge \sigma \precsim_{\mathsf{m}} \sigma') \Rightarrow \neg(\sigma' \preceq_{\mathsf{m}} \sigma \wedge \sigma \preceq_{\mathsf{m}} \sigma') \qquad \text{(by conversion)}$$
$$\sigma' \precsim_{\mathsf{m}} \sigma \wedge \sigma \precsim_{\mathsf{m}} \sigma' \Leftarrow \sigma' \preceq_{\mathsf{m}} \sigma \wedge \sigma \preceq_{\mathsf{m}} \sigma' \qquad \text{(by Lemma 4) } \square$$

In Sect. 7.6, we revisit how features such as partial orders account for uncertainty in RISKSTRUCTURES.

5.2. Quantitative mitigation orders

So far, we have seen how partial orders account for a lack of knowledge and potential uncertainties about risk. Now, we will investigate the use of *impact or consequence* data in form of severity intervals, if available for specific factors, to interpolate knowledge gaps, model uncertainty, and derive a linear order over R.

We continue with definitions for dealing with intervals. Given two intervals $[l_1, u_1), [l_2, u_2) \subset \mathbb{R}_+$, the *convex hull* is a map $\sqcup\colon \mathbb{R}_+^2 \times \mathbb{R}_+^2 \to \mathbb{R}_+^2$ given by

$$[l_1, u_1) \sqcup [l_2, u_2) = [\min\{l_1, l_2\}, \max\{u_1, u_2\}) \,. \tag{4}$$

For a family of $n \in \mathbb{N}$ intervals $I = ([l_i, u_i))_{i\in[1..n]}$, we use the abbreviation $\bigsqcup I = [l_1, u_1) \sqcup \ldots \sqcup [l_n, u_n)$. Furthermore, let $active\colon R(F) \to 2^F$ with $active(\sigma) = \{\mathsf{f} \in F \mid \sigma(\mathsf{f}) = f\}$ be the map returning the set of active factors of a risk state. Moreover, let $S\colon R \to \mathbb{R}_+^2$ with

$$S(\sigma) = \bigsqcup (\mathsf{f}.s)_{\mathsf{f}\in active(\sigma)} \tag{5}$$

be a map for the construction of the severity interval of a risk state from the intervals of its factors.[15] Whereas the *minimal severity* of a factor f is given by $\mathsf{f}.s = [0, 0)$, the minimal severity $S(\sigma)$ of a state σ is the empty interval $[)$, which we equate with the empty set, $[) = \varnothing$. For any two real-valued intervals $[a, b), [c, d) \in \mathbb{R}_+^2$, Ishibuchi and Tanaka [IT90] define with $[a, b) \leq [c, d) \iff a \leq c \wedge b \leq d$ a partial order over such intervals.

Moreover, we say that two risk states $\sigma, \sigma' \in R$ are *severity-equivalent* if and only if their accumulated severity intervals are equal, that is, $\sigma \sim_{\mathsf{s}} \sigma' \iff S(\sigma) = S(\sigma')$. We have that $\sigma = \sigma' \Rightarrow \sigma \sim_{\mathsf{s}} \sigma'$ because the factors that are in their active phases are identical. The relation $\sim_{\mathsf{s}}$ is an equivalence relation because it is reflexive, symmetric, and transitive (all by the usual equivalence over intervals). Furthermore, $\sim_{\mathsf{s}}$ induces equivalence classes $[\sigma]_{\sim_{\mathsf{s}}} = \{\sigma' \in R \mid \sigma' \sim_{\mathsf{s}} \sigma\}$ over R for any $\sigma \in R$ with the corresponding quotient class $R/\sim_{\mathsf{s}}$. With the family $(\mathsf{f}.s)_{\mathsf{f}\in F}$ of *severity intervals* of F, we now define an order over $R/\sim_{\mathsf{s}}$.

Definition 10 (Strong mitigation order) For $[\sigma]_{\sim_{\mathsf{s}}}, [\sigma']_{\sim_{\mathsf{s}}} \in R/\sim_{\mathsf{s}}$, define

$$[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}} \iff \forall\, \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}}\colon S(\check{\sigma}) \geq S(\check{\sigma}') \vee S(\check{\sigma}') \subset S(\check{\sigma}) \,.$$

$[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$ can be dropped from $R/\sim_{\mathsf{s}}$, yielding

$$\forall\, \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}}\colon \check{\sigma} \leq_{\mathsf{m}} \check{\sigma}' \iff S(\check{\sigma}) \geq S(\check{\sigma}') \vee S(\check{\sigma}') \subset S(\check{\sigma}) \,. \tag{6}$$

$\leq_{\mathsf{m}}$ codifies that σ' *(a) reduces or (b) focuses risk* if the union of its severity intervals is

1. *Lower* in the ranking $\leq$ of interval numbers or
2. *Strictly narrower* than the corresponding union for σ.

Condition (a) seems immediately intuitive. Condition (b) conveys the intuition that the interval carries less uncertainty about the consequences expected from σ' than from σ. Equivalence classes in $R/\sim_{\mathsf{s}}$ abstract from the factors from which the merged severity intervals originate. This abstraction has to be carefully taken into account when using $\leq_{\mathsf{m}}$ and, therefore, when specifying severity. Note that $\leq_{\mathsf{m}}$ is based on the convex hull of severity intervals from the active phases of a pair of risk states. Apart from the convex hull, interval addition and multiplication are relevant for alternative mitigation orders as we shall see below. However, a detailed investigation is left for future work. Let us now consider some properties of $\leq_{\mathsf{m}}$.

[15] Note that S over-approximates (i.e., constructs the convex hull from) sparsely distributed severity intervals.

Lemma 5 $\preceq_{\mathsf{m}}$ is linear over $R/{\sim_{\mathsf{s}}}$.

Proof sketch. We show by case analysis that any two risk states are comparable and $\preceq_{\mathsf{m}}$ is antisymmetric. The complete proof is stated in Appendix A. □

Corollary 3 After dropping Lemma 5 by Formula (6), we have that $\preceq_{\mathsf{m}}$ is also linear over R.

Lemma 6 $(R, \preceq_{\mathsf{m}})$ and $(R/{\sim_{\mathsf{s}}}, \preceq_{\mathsf{m}})$ are well ordered.

Proof sketch. Lemma 6 follows from a finite R (by definition) and, thus, finite $R/{\sim_{\mathsf{s}}}$, and linearity of $\preceq_{\mathsf{m}}$ (by Lemma 5). □

Definition 11 For $\sigma \in R(F)$, we also write $\mathbf{0}^F \equiv \forall\, \mathsf{f} \in F\colon \sigma(\mathsf{f}) = 0^f$ and $\mathbf{F} \equiv \forall\, \mathsf{f} \in F\colon \sigma(\mathsf{f}) = f$. We denote by $\top^F$ the set of *maximal elements* and by $\bot^F$ the set of *minimal elements* of $(R, \preceq_{\mathsf{m}}, \precsim_{\mathsf{m}}, \leq_{\mathsf{m}})$. We characterise the minimal elements in $(R(F), \preceq_{\mathsf{m}})$ by

$$\bot^F \equiv \{\sigma \in R(F) \mid \forall\, \sigma' \in R(F)\colon \sigma' \preceq_{\mathsf{m}} \sigma \Rightarrow \sigma' =_{\mathsf{m}} \sigma\}$$

and analogously for $(R, \precsim_{\mathsf{m}})$ and $(R, \leq_{\mathsf{m}})$ and the maximal elements.

Corollary 4 If $\forall\, \mathsf{f} \in F\colon (Ph_{\mathsf{f}}, \preceq_{\mathsf{f}})$ is linear, then $(R(F), \preceq_{\mathsf{m}}) = (R(F), \precsim_{\mathsf{m}})$. If $R \neq R(\varnothing)$ then $\bot^F$ and $\top^F$ are non-empty and, therefore, have a proper manifestation. For $(R/{\sim_{\mathsf{s}}}, \leq_{\mathsf{m}})$, $\bot^F$ and $\top^F$ are singletons.

Proof of Corollary 4. The proof is by contradiction. For the sake of brevity, we only consider a sketch of this proof. Assume we have two state classes $[\sigma]_{\sim_{\mathsf{s}}}, [\sigma']_{\sim_{\mathsf{s}}}$ in $\bot^F$ with $[\sigma]_{\sim_{\mathsf{s}}} \neq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$. Because of our assumption, state classes are in linear order. Thus, by definition of $\bot^F$, one of these state classes causes a violation of the universal quantification in Definition 11 and, therefore, one of the classes cannot be in $\bot^F$ which contradicts our assumption. The proof is analogous for $\top^F$. □

Corollary 5 $(R/{\sim_{\mathsf{s}}}, \leq_{\mathsf{m}})$ forms a complete lattice.

Proof of Corollary 5. Linearity of $\leq_{\mathsf{m}}$ implies that every non-empty subset of $R/{\sim_{\mathsf{s}}}$ has a greatest lower bound and a least upper bound. □

5.3. Relating mitigation orders

The strong mitigation order characterised by the Lemmas 5 and 6 is driven by the number of active factors and their severity intervals (because of the definition of S) but not by the equality of factor phases among the compared risk states. This offers the possibility of abstraction from individual factors and focusing on severity estimates. To avoid infeasible models (e.g. specifications that get too strong to be realisable), we *require* that the addition of severity intervals constitutes a *relational extension* of either $\preceq_{\mathsf{m}}$ or $\precsim_{\mathsf{m}}$, formally,

$$\left(\forall\, \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}}\colon \check{\sigma} \preceq_{\mathsf{m}} \check{\sigma}' \vee \check{\sigma} \precsim_{\mathsf{m}} \check{\sigma}'\right) \Rightarrow [\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}\,.$$

Dropped to R, this implies $\check{\sigma} \preceq_{\mathsf{m}} \check{\sigma}' \vee \check{\sigma} \precsim_{\mathsf{m}} \check{\sigma}' \Rightarrow \check{\sigma} \leq_{\mathsf{m}} \check{\sigma}'$ for all pairs $(\check{\sigma}, \check{\sigma}') \in [\sigma]_{\sim_{\mathsf{s}}} \times [\sigma']_{\sim_{\mathsf{s}}}$, therefore,

$$\check{\sigma} \preceq_{\mathsf{m}} \check{\sigma}' \vee \check{\sigma} \precsim_{\mathsf{m}} \check{\sigma}' \Rightarrow S(\check{\sigma}) \geq S(\check{\sigma}') \vee S(\check{\sigma}') \subset S(\check{\sigma}) \tag{7}$$

for full and partial comparability, otherwise implying

$$(\check{\sigma}, \check{\sigma}') \notin \preceq_{\mathsf{m}} \wedge\ (\check{\sigma}, \check{\sigma}') \notin \precsim_{\mathsf{m}} \Rightarrow \mathrm{T}\,. \tag{8}$$

Intuitively, if $\check{\sigma}$ is "worse" than $\check{\sigma}'$ then its accumulated severity interval $S(\check{\sigma})$ has to be greater than that of $\check{\sigma}'$ and, therefore, must not be contained in that of $\check{\sigma}'$. Moreover, if $\check{\sigma}$ and $\check{\sigma}'$ are incomparable in $\preceq_{\mathsf{m}}$ and $\precsim_{\mathsf{m}}$ (i.e., some factors have inversely ordered or incomparable phases) then $S(\check{\sigma})$ and $S(\check{\sigma}')$ are allowed to form any relationship (signified by T for "true"), for example, Formula (7).

What is the (necessary and) *sufficient condition* on F to satisfy the requirement expressed by Formula (7)? Risk spaces and risk state pairs are the interpretations and, therefore, potential models satisfying the relational extension imposed by Formula (7). Answering this question suggests the following lemma.

Lemma 7 For $\sigma, \sigma' \in R(F)$,

$$\sigma \preceq_{\mathsf{m}} \sigma' \vee \sigma \precsim_{\mathsf{m}} \sigma' \Rightarrow \mathit{active}(\sigma) \supseteq \mathit{active}(\sigma')\,.$$

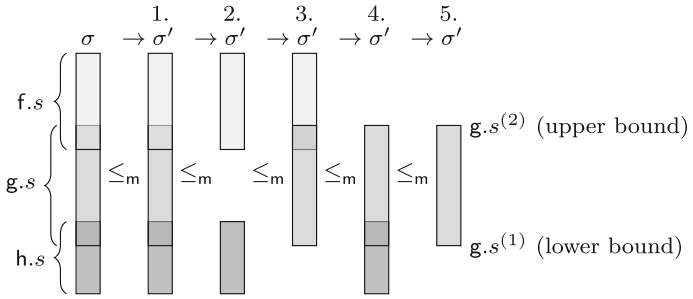


Fig. 5. Visualisation of the case distinction to establish Theorem 1, with $\mathsf{f}, \mathsf{g}, \mathsf{h} \in \mathcal{F}$, TS $(R, \rightarrow)$, and $\sigma \rightarrow \sigma'$

Proof sketch. The proof is by induction over F and relies on the assumptions that, for any $\mathsf{f} \in F$, f is the *unique maximal element* in $\preceq_{\mathsf{f}}$ and that the map *active* (Sect. 5.2) only returns such elements. The whole proof is stated in Appendix A. □

Again, fix a finite F and a pair $\sigma, \sigma' \in R(F)$ and assume $\sigma \preceq_{\mathsf{m}} \sigma' \vee \sigma \precsim_{\mathsf{m}} \sigma'$. Then, by Lemma 7, σ' incorporates *a subset of* σ's active factors. To show that $S(\sigma')$ preserves $S(\sigma') \subset S(\sigma)$, the right-hand part of the disjunction in the consequent of Formula (7), it is sufficient to investigate $S(\sigma')$ for any factor deactivation possible in a transition from σ to σ'. We can summarise all such possibilities in five cases:

1. all intervals remain ($\sigma' = \sigma$),
2. only intervals in the convex hull of the remaining intervals are removed,
3. intervals only increasing the lower bound of this hull are removed,
4. intervals only decreasing the upper bound of this hull are removed, and
5. intervals increasing the lower bound and decreasing the upper bound of this hull are removed.

This case distinction is visualised in Fig. 5 and also works for σ' such that $\sigma \prec_{\mathsf{m}} \sigma'$. Formula (7) is also satisfied if all factors in F are assigned the same interval, call it s^F. In conclusion, the sufficient condition on F to satisfy Formula (7) is the "unique maximal element" precondition in the proof of Lemma 7. Apart from this precondition, Formula (7) holds of an arbitrary finite $F \subseteq \mathcal{F}$. Below, we shall call $\preceq_{\mathsf{m}}$ and $\precsim_{\mathsf{m}}$ *inclusive mitigation orders*, and $\leq_{\mathsf{m}}$ a *strong mitigation order*. We arrive at the following theorem.

Theorem 1 The *strong mitigation order* $\leq_{\mathsf{m}}$ extends the *partially comparable inclusive mitigation order* $\precsim_{\mathsf{m}}$ which, in turn, extends the *fully comparable inclusive mitigation order* $\preceq_{\mathsf{m}}$. Formally, for $\sigma, \sigma' \in R$:

$$\sigma \preceq_{\mathsf{m}} \sigma' \overset{\text{Lemma 4}}{\Longrightarrow} \sigma \precsim_{\mathsf{m}} \sigma' \overset{\text{Lemma 7}}{\Longrightarrow} \sigma \leq_{\mathsf{m}} \sigma'.$$

5.4. Application: evaluating local, regional, and global safety

The orders $\preceq_{\mathsf{m}}$, $\precsim_{\mathsf{m}}$, and $\leq_{\mathsf{m}}$ allow a *local* evaluation of safety in the sense that their definitions only require the comparison of pairs of risk states. Two further qualitative notions of safety seem to be useful.

Let R be non-empty and finite and $reach\colon R \times \mathcal{P} \to 2^R$. Given a process $P \in \mathcal{P}$ and a risk state $\sigma \in R$, $reach(\sigma, P) \subseteq R$ denotes the set of risk states reachable from σ by finite executions (prefixes) of P where σ itself is always reachable and, thus, $\sigma \in reach(\sigma, P)$. Figure 6 (light-grey area) exemplifies such a set. Then, we use the presented orders to determine non-empty sets of minimal and maximal elements in R reachable by P, such as $\max_{\preceq_{\mathsf{m}}} reach(\sigma, P)$ or $\min_{\precsim_{\mathsf{m}}} reach(\sigma, P)$. In Fig. 6, the risk states 1 to 3 and 5 to 7, connected by double arcs, are the ones reachable from σ after at most two consecutive endangerments or mitigations. Reachability arcs are labelled with the weakest applicable order. Single arcs illustrate potential modifications or extensions of the reachability set, for example, by the states 4 and 8 to 10 after a reassessment of the operational environment by the machine.

In the situation described by the process P in the risk state σ, these two sets signify the *regionally safest* (max, white circles) and the *regionally most hazardous* (min, dark-grey circles) states, respectively. The smallest such set will only and exactly contain σ, meaning P cannot reduce risk in $(R, \precsim_{\mathsf{m}})$. $\preceq_{\mathsf{m}}$ and $\precsim_{\mathsf{m}}$ enable a *regional* evaluation of safety inasmuch as once a maximal element in $\max_{\precsim_{\mathsf{m}}} reach(\sigma, P)$ is reached, $\precsim_{\mathsf{m}}$ limits reasoning about safer states that P could reach from σ.

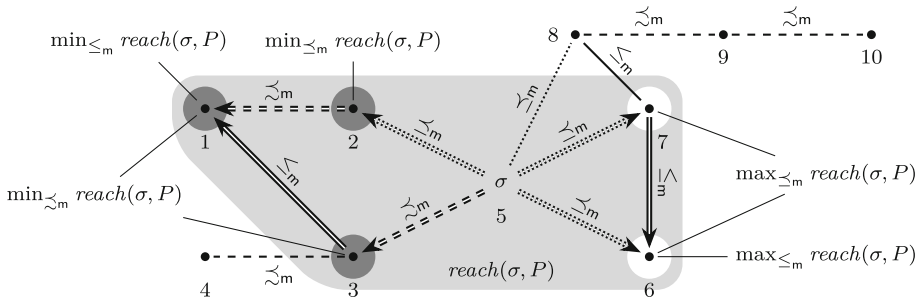


Fig. 6. Visualisation of local, regional, and global safety in a risk space R

Besides local and regional safety, $(R/\sim_s, \preceq_m)$ supports *global* evaluation because of its linearity (Lemma 5). With $\preceq_m$, there are always *unique* safest and riskiest states in $reach(\sigma, P)/\sim_s$ (Corollaries 4 and 5). In contrast, $\preceq_m$ and $\precsim_m$ will not guarantee this uniqueness. Note that the use of equivalence classes leads to more abstract forms of safest and riskiest states. Overall, Lemma 6 and Corollary 5 provide necessary conditions for deriving *finite strategies* (i.e., policies, choice resolutions) that stabilise or terminate P in a safest state.

Quantitative mitigation orders (e.g. $\preceq_m$) will have to be calculated at run-time, according to $\mathfrak{R}$ by estimating probabilities of factor occurrence and severity of factor consequences from situational data only available during operation. In P, *safety* can then be observed as the *gradual presence or absence of risk* over $traces(P)$. On a side note, the *reliability* as the *gradual presence or absence of defective behaviour* can be seen as a special case when considering only factors that model faults.

5.5. Discussion: ethical aspects of mitigation orders

Linear mitigation orders such as $\preceq_m$ promote machines with *negative utilitarian* decision ethics [War12, p. 51]. For example, severity intervals could be calculated at run-time based on sensor data about the possible operational situation of the machine. Expected outcomes of enabled mitigations, if any, will then be comparable according to $\preceq_m$. This comparability allows the assessment of the actual reachability of states with strictly lower risk. Any resolution of a near-accident situation (Sect. 4.3) or a *tram*[16] *problem* [Foo78] would then consist in the choice of the mitigation leading to the state with the lowest risk or *the least severe of the expected negative outcomes*. This scheme characterises negative utilitarianism.

Linear mitigation orders globally resolve decisions based on explicit and, therefore, disputable criteria. Consequently, utilitarian ethics have been criticised to lead to oversimplified approaches to resolve indecision. Such critiques stress the difficulty of predicting the positive and negative effects (i.e., "double effects" [Foo78]) of certain actions, in our case, the estimation of the severity of an activated factor and the reached state [War12, pp. 48-49]. Appreciating the equality of any two groups of the human family [Uni48], one could conclude that tram problems should be solved by random decisions, independent of whether these are made by humans or machines. However, in analogy to risk-averse humans (Sect. 3), a risk-aware machine should make (self-inflicted) tram problems unlikely, that is, avoid such decisions rather than aiming at their random or utilitarian resolution. The likelihood of tram problems is an important subject to be investigated.

Accordingly, a structured risk model with dependencies between factors could be used to complement utilitarian decision ethics with Kantian ethics, that is, to use $\preceq_m$ to decide about conservative measures *before* high-severity factors get activated. This results in what is called "rule utilitarianism" [War12, p. 52]. For example, we can model the *necessary preconditions* of certain tram problems as risk factors and a machine based on this model could use these factors to constrain its behaviour. Although the presented model can be used with linear orders, the discussions below stay agnostic of the mitigation order.

[16] Also known as the "trolley problem", a situation where any of the enabled actions leads to an unacceptable outcome.

6. Factor dependencies

Usually, only subsets of a risk space R and a process P's traversals of R are relevant. Identifying these subsets is the task of risk analysis and can be difficult if the relevance of each risk state can only be determined at run-time based on its context in R and the state of P. As highlighted in Sect. 3, it is then useful to model *(causal) relationships* between (phases of) factors, their events (i.e., activation, mitigation), and, consequently, risk states. For example, we might want to specify that (i) the *activation* of a risk factor *causes* the *activation* of another factor, (ii) the *mitigation* of one factor *causes* the *activation* of several other factors, or (iii) the *activation* of a factor *requires* the *activation* of some other factors. We can take account of dependencies, such as illustrated in Table 2 (e.g. `requires` ncol, `causes` $\mathsf{loc_d}$), by imposing *constraints* on pairs of consecutive risk states and their comprising factors' phases. For this, we use binary relations over R to hypothesise *causality assumptions* about less known or controllable parts of P (i.e., the environment En) and express *causality requirements* for known and controllable parts of P (i.e., the system Sy).

6.1. Relations over risk spaces

Let $F \subseteq \mathcal{F}$ and consider the space $R(F)$ and two distinct factors $\mathsf{f}, \mathsf{g} \in F$. In the following, we employ relational specification to formalise factor dependencies as relations over R, that is, as subsets of $R \times R$. For example, a `causes`[17] constraint requires of a pair $(\sigma, \sigma') \in R \times R$ that if f is active then, within at most one (logical) step, g must be active until f gets either inactive or mitigated.

Let $\mathcal{C}$ be the set of all constraints. For the translation of constraints into a form useful for the discussion below, we use the map $[\![\cdot]\!]^R_{\mathsf{c}} \colon \mathcal{C} \to 2^{R \times R}$ to denote the relational semantics of constraints over R. To distinguish different semantic maps, as will be defined later, we use subscripts, for example, c in $[\![\cdot]\!]^R_{\mathsf{c}}$. Recall that we use $\sigma(\mathsf{f})$ to refer to the phase of factor f in state σ (Sect. 4.2).

For example, `causes` constraints can be encoded as relations over R by the definition

$$[\![\mathsf{f}\ \texttt{causes}\ \mathsf{g}]\!]^R_{\mathsf{c}} = \{(\sigma,\sigma') \in R \times R \mid \underbrace{\sigma(\mathsf{f}) \neq f \wedge \sigma'(\mathsf{f}) = f}_{\text{activation of f...}} \wedge \underbrace{\sigma(\mathsf{g}) \neq \overline{g}}_{\text{when g not mitigated}} \Rightarrow \underbrace{\sigma'(\mathsf{g}) = g}_{\text{...activates g}}\} \,. \tag{9}$$

This constraint implies that factor activation takes a logical time step corresponding to a strictly positive real-time duration (i.e., > 0) in P. Note that this definition of `causes` encodes the assumption that g, once or as long as mitigated, cannot be caused (again) by activating f. While this behaviour is often to be expected of any mitigation designed for g, one can define a more general constraint, say `alwaysCauses`, to capture this behaviour. Moreover, `causes` is to be read as "is sufficient to cause" rather than as "is necessary to cause", and it extends to factor sets. Given $F, G \subseteq \mathcal{F}$ with $F \cap G = \varnothing$, we define

$$[\![F\ \texttt{causes}\ G]\!]^R_{\mathsf{c}} = \{(\sigma,\sigma') \in R \times R \mid \exists \mathsf{f} \in F\, \forall \mathsf{g} \in G \colon \sigma(\mathsf{f}) \neq f \wedge \sigma'(\mathsf{f}) = f \wedge \sigma(\mathsf{g}) \neq \overline{g} \Rightarrow \sigma'(\mathsf{g}) = g\} \,.$$

Note that all pairs (σ, σ') violating the antecedent of the conditional are in $[\![\mathsf{f}\ \texttt{causes}\ \mathsf{g}]\!]^R_{\mathsf{c}}$ as well.

As another example, the `requires` constraint can be defined in relational form by

$$[\![\mathsf{f}\ \texttt{requires}\ \mathsf{g}]\!]^R_{\mathsf{c}} = \{(\sigma,\sigma') \in R \times R \mid \underbrace{\sigma'(\mathsf{f}) = f}_{\text{an active f requires...}} \Rightarrow \underbrace{\sigma(\mathsf{g}) = g}_{\text{an active g in the preceding state}}\} \,. \tag{10}$$

The lifting of `requires` to factor sets $F, G \subseteq \mathcal{F}$ with $F \cap G = \varnothing$ is described as

$$[\![F\ \texttt{requires}\ G]\!]^R_{\mathsf{c}} = \{(\sigma,\sigma') \in R \times R \mid \exists \mathsf{f} \in F \colon \sigma'(\mathsf{f}) = f \Rightarrow \forall \mathsf{g} \in G \colon \sigma(\mathsf{g}) = g\} \,.$$

This variant of the `requires` constraint refers to *all* factors specified on its right-hand side and, this way, resembles an `AND`-gate as used in FTA. The side condition $F \cap G = \varnothing$ avoids that a factor requires or causes itself (i.e., it avoids the "chicken and egg" problem).

[17] Used in form of MCSs in FTA and, less frequently formally, in FMEA (see Sect. 2.1).

Table 3. A selection of constraints more commonly used in causal reasoning

Name	PC	D	P	C	M	FC	Y	Description
`causes`	$F \rightleftarrows G$	$\rightarrow$	+	w	1:n	=n	✓	The *activation* of a factor in F *causes* the (successive) *activation* of the factor set G, Formula (9).
`causes`$^{-1}$	$\overline{F} \rightleftarrows G$	$\rightarrow$	+	w	m:n	=n	-	The *mitigation* of the factor set F *causes* the *activation* of the factor set G.
`requires`	$F \rightleftarrows G$	$\leftarrow$	+	s	1:n	=n	✓	The *activation* of a factor in F *requires* the (prior) *activation* of all factors in G. `AND`-gate in FTA, Formula (10).
`requires`$_1$	$F \rightleftarrows G$	$\leftarrow$	+	s	1:n	$\geq$1	✓	The *activation* of a factor in F *requires* the (prior) *activation* of at least *one* factor in G; `OR`-gate in FTA.
`prevents`	$F \rightleftarrows G$	$\rightarrow$	–	s	1:n	=n	✓	The *activation* of a factor in F *prevents* or *impedes* the *activation* of all factors in G.
`preventsMit`	$F \rightleftarrows \overline{G}$	$\rightarrow$	–	s	1:n	=n	✓	The *activation* of a factor in F *prevents* or *impedes* the *mitigation* of all factors in G.
`excludes`	$F \rightleftarrows 0^G$	$\rightarrow$	+	s	1:n	=n	✓	The *activation* of a factor in F *deactivates* (superposes or invalidates) all factors in G.
`direct`	$F \rightleftarrows 0^G$	$\rightarrow$	+	s	1:0	=1	✓	The *deactivation* of a factor in F *must* only happen directly from *its activated phase*.
`offRepair`	$F \rightleftarrows 0^G$	$\rightarrow$	–	s	1:0	=1	✓	The *deactivation* of a factor in F *may not* happen directly from *its activated phase*.

Legend: See the *dimensions* discussed in the text of Sect. 6.1; ✓ . . . implemented in YAP [Gle21]

Remark 1 `causes` models weak causality and `requires` models a strong causality. `causes` and `requires` constitute basic dependency templates and exemplify how dependencies support the safety engineer or risk analyst in causal modelling in a state-based relational way.

Analogously, there are many possible *factor dependencies* over R that resemble causal reasoning of techniques such as FTA (Sect. 2.1). Such dependencies can be classified along several dimensions:

- *phase combination* (PC): active to active ($F \rightleftarrows G$), active to mitigated ($F \rightleftarrows \overline{G}$), active to inactive ($F \rightleftarrows 0^G$), mitigated to mitigated ($\overline{F} \rightleftarrows \overline{G}$), mitigated to active ($\overline{F} \rightleftarrows G$), mitigated to inactive ($\overline{F} \rightleftarrows 0^G$); combinations thereof are possible;
- *direction* (D) of cause-effect analysis: forward ($\rightarrow$, e.g. for modelling sufficient conditions or propagation), backward ($\leftarrow$, e.g. for modelling necessary conditions or explanation);
- *polarity* (P): obligation (+), permission ($\circ$), inhibition ($-$);
- *causality* (C): strong (s, sequential events), weak (w, simultaneous events);
- *multiplicity* (M): one-to-one ($1:1$), one-to-many ($1:n$), many-to-one ($n:1$), many-to-many ($m:n$) where $m, n > 0$; self-referential[18] ($1:0$);
- *factor combination* (FC): "$\sim m$ out of n" where $\sim \in \{\leq, =, \geq\}$ and $1 \leq m \leq n$.

These dimensions[19] characterise factor dependencies commonly used in risk analysis. Most of the constraints listed in Table 3 are realised in our tool YAP [Gle21], which enables their use based on previous discussions in [Gle18, Gle17]. However, their comprehensive treatment would exceed the scope of this work.

We have now seen that constraints prune irrelevant state pairs. This mechanism is reflected by the following definition.

Definition 12 (Relational semantics of constraints) For a set of constraints $C \subseteq \mathcal{C}$,

$$[\![C]\!]^R_{\mathsf{c}} = \begin{cases} R \times R, & C = \varnothing \\ \bigcap_{k \in C} [\![k]\!]^R_{\mathsf{c}}, & \text{otherwise} \end{cases} \quad \subseteq R \times R\,. \tag{11}$$

Constraints are an instrument for specifying structures over risk spaces. Regarding the composition of spaces, this instrument requires the definition of the following well-formedness condition.

[18] In this case, we allow $F = G = \{\mathsf{f}\}$.

[19] Naïve combination of all dimensions results in more than 1000 possibilities to define constraints. Many of these possibilities are not essentially different and some might not even be useful. Table 3 does, however, not aim to cover all useful constraints.

Listing 1: YAP script modelling the situation `transferObj` from Table 4

```
factormodel {
  dmgo desc "object damage"
    mishap
    requiresNOf(1|hp,fodg)
    sev=[5,10);
  fodg desc "failure on demand of grabber"
    requires(slp)
    causes(dmgo);
  slp desc "slippery grabber"
    requires(wet)
    direct
    detectedBy(.senseSlip)
    mitigatedBy(.incPr)
    sev=[2,6);
  wet desc "wet grabber surface"
    causes(slp)
    direct;
  hp desc "high grabber pressure"
    mitPreventsMit(slp)
    direct
    detectedBy(.sensePr)
    mitigatedBy(.limPr)
    sev=[3,8);
  /* str desc "spurious release"; equal to fod */
}

controlloop collRob {
  mode senseSlip desc "identify slip";
  mode incPr desc "increase grabbing pressure"
    update "pressure++";
  mode sensePr desc "sense pressure"
    guard "pressure > threshold(obj)";
  mode limPr desc "limit grabbing pressure"
    update "pressure = threshold(obj)";
}
```

Table 4. Situation/action/factor table for grabbing scenarios performed by collaborative robots

	Act.	work (w)	grab & hold (g)				release (r)
Situ.	**Fact.**	$\mathsf{dmg_o}$ object damage	$\mathsf{fod_g}$ failure on demand	slp slippery grabber	wet wet surface	hp high pressure	$\mathsf{st_r}$ spurious trip
`transferObj`		3. `requires`$_1$ hp, fod_g	3. `requires`$_1$ slp, $\mathsf{st_r}$; causes $\mathsf{dmg_o}$	2. $Pr[\underline{slp} \mid slp] = h$ 3. `requires` wet 4. increase grabbing pressure	3. `causes` slp	2. $Pr[\underline{hp} \mid hp] = m$ 3. `mitPrevents-Mit` slp 4. limit grabbing pressure	–

Legend: h... high, m... medium, –not analysed; see also Table 2 for guidance

Definition 13 (Well-formedness of constraints) Let $R(F)$ be a space formed by a factor set $F \subseteq \mathcal{F}$. We say that a constraint $k \in \mathcal{C}$ is *well-formed for* $R(F)$ if and only if it does not refer to factors other than the ones in F, formally, if and only if $[\![k]\!]^R_{\mathsf{c}} \subseteq R(F) \times R(F)$. We say that $C \subseteq \mathcal{C}$ is well-formed for $R(F)$ if and only if each element in C is well-formed for $R(F)$.

6.2. Example: collaborative robots

To illustrate relationships for structuring risk perception, we apply our framework to a scenario in human-robot collaboration. Table 4 shows a risk analysis similar to Table 2 for a robotic arm with a grabber. Consider such a robot *(w)orking* by repetitively *(g)rabbing, holding*, and *(r)eleasing* work pieces. Water or oil on the robot's grabber (wet) can *cause* the grabber to be *slippery* (slp) such that *holding* an object may increase the likelihood of *accidentally dropping* it ($Pr[\underline{slp} \mid slp]$). For our further analysis, we generalise the negative outcome from the phase $\underline{slp}$ to an additional *final* risk factor $\mathsf{dmg_o}$. Obviously, increased *grabbing pressure* can mitigate slp. However, forward reasoning leads to the conclusion that increased pressure could be *too high* (hp) *causing* damage to the object ($\mathsf{dmg_o}$). However, because we plan to install an *intervention* in our machine, we omit the dependency hp `causes` $\mathsf{dmg_o}$. Moreover, a FTA of the sensor software and the actuator hardware results in the factor *mistaken loosening* of the grabber ($\mathsf{st_r}$), which, for the sake of simplicity, we equate with a failure on demand of the "grab & hold" action ($\mathsf{fod_g}$). Applying final backward causal reasoning, the object's high *falling* onto a hard surface `causes` damage of the object ($\mathsf{dmg_o}$) and `requires`$_1$ (at least one of) slippery grabber (slp), $\mathsf{fod_g}$, or a spurious trip of the "release" action ($\mathsf{st_r}$).

Listing 1 encodes this analysis in a script processable by YAP, which calculates the risk space and the phase transition relation. The resulting risk graph in Fig. 7 can be useful as a controller design template for a risk-aware machine. Further details on how this graph can be constructed are discussed in Sect. 7.

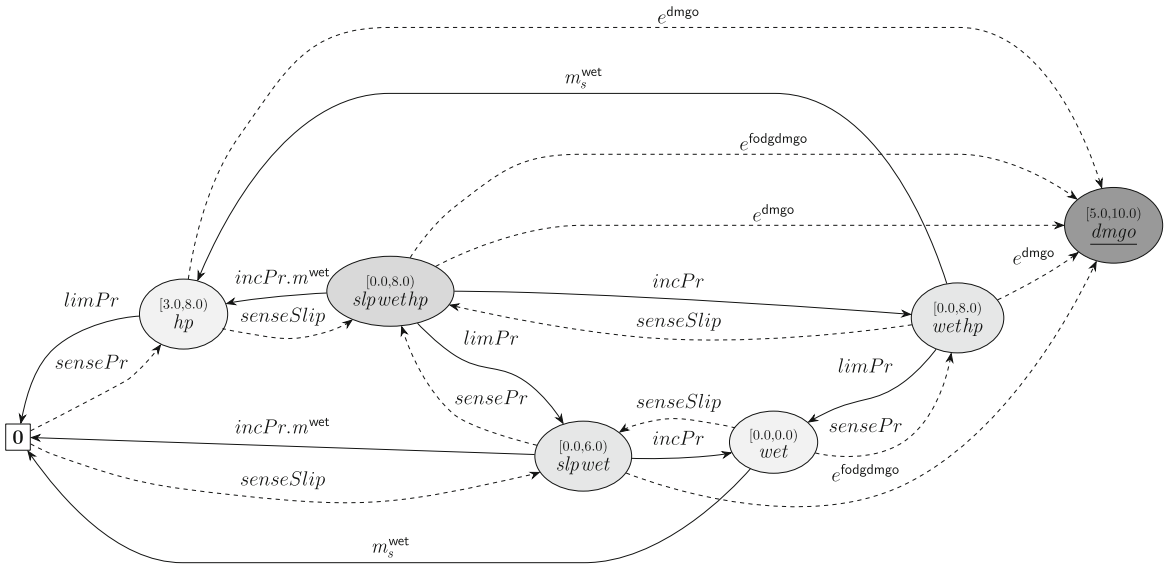


Fig. 7. Graph for risk structure $\mathfrak{R}_{\mathsf{transferObj}}$ with factor set $F = \{\mathsf{dmg}, \mathsf{fod}, \mathsf{slp}, \mathsf{wet}, \mathsf{hp}\}$ generated by YAP from Listing 1. To increase readability, inactive factors are not shown in the nodes

6.3. Discussion: abstraction, compatibility, and characteristics of risk factors

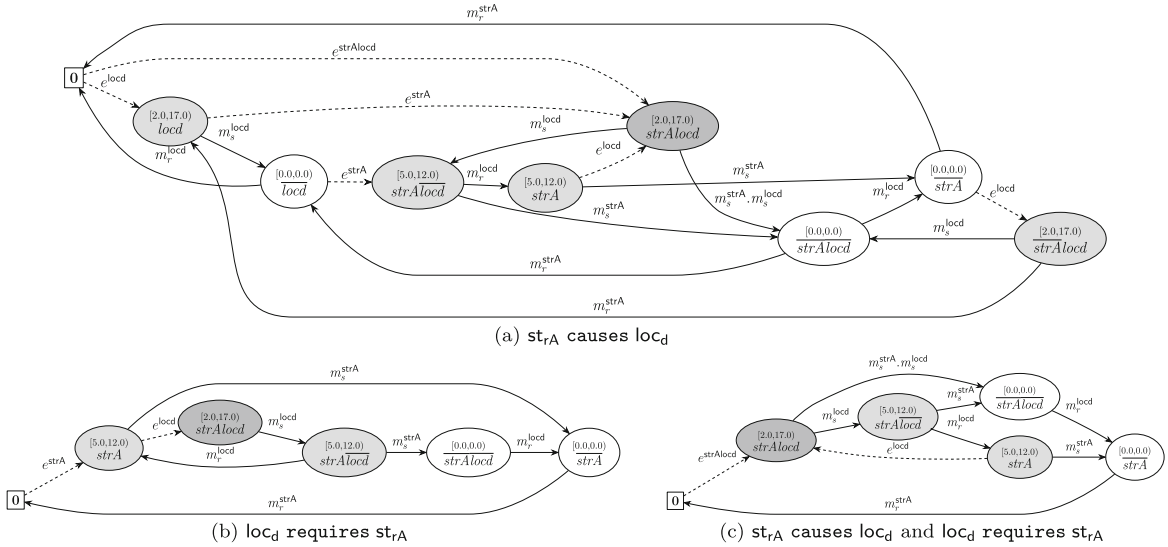
Abstraction Two structurally different risk states $\sigma, \sigma' \in R$ (i.e., $\sigma \neq \sigma'$) can be severity-equivalent (i.e., $\sigma \sim_{\mathsf{s}} \sigma'$). Moreover, the family $(\mathsf{f}.s)_{\mathsf{f} \in F}$ of severity intervals of F forms a *cut of causal chains* and, thus, defines the *scope* of $\mathfrak{R}$. These intervals abstract from potential *consequences of active* factors. This abstraction is left to the modeller (i.e., the risk analyst or safety engineer) and can vary significantly, depending on the *assets* (e.g. humans, animals, environment) and *impact categories* (e.g. certain injuries, damage, loss) under discussion. For example, assume two independent consequences c_A and c_B (e.g. a class of injuries of an operator A, a class of damages of an object B) associated with *possible*[20] intervals $[l_A, u_A)$ and $[l_B, u_B)$. Consider a *single* factor f causing both c_A and c_B. This circumstance suggests the consistency condition $\mathsf{f}.s \subseteq [l_A + l_B, u_A + u_B)$, requiring that all consequences of f (i.e., c_A, c_B) have to be measurable in their severity along the *same scale*. Our example stresses the difficulty of comparing injury with damage. However, if c_B was another class of injuries of that operator then c_B would fit into the same scale as c_A. Practitioners often formalise these concepts for quantitative analysis. However, we leave this for future work.

Factor compatibility Consequences of *several* factors should be *compatible* such that the convex hull of their intervals (recall $\sqcup$ from Formula (4)) has a consistent meaning of severity in $(R, \leq_{\mathsf{m}})$. Consider the two factors f and g. Both can describe three cases with their *individually specified* intervals $\mathsf{f}.s$ and $\mathsf{g}.s$:

1. f and g share all their consequences (e.g. both cause c_A). f, g → c_A
2. f and g share some of their consequences (e.g. f causes c_A and c_B, g causes c_B). f → c_A c_B ← g
3. f and g do not share any consequences (e.g. f causes c_A, g causes c_B). f → c_A g → c_B

If both factors get active in case 1, the convex hull can be backed by the condition $\mathsf{f}.s \sqcup \mathsf{g}.s \subseteq [l_A, u_A)$. Case 2 can be split into the $\leq_{\mathsf{m}}$-compatible case 1 for c_B and case 3 for c_A. In case 3, generally, the convex hull may extend both the range of consequences and the severity intervals. However, the consistency condition $\mathsf{f}.s \sqcup \mathsf{g}.s \subseteq [l_A, u_A) \sqcup [l_B, u_B)$ implied by $\leq_{\mathsf{m}}$ seems inappropriate. For example, if c_A and c_B signify damage of two independent objects with the same interval, the hull would not account for this because of idempotency of interval union.

[20] For $[l_A, u_A)$ and $[l_B, u_B)$, we assume sufficient knowledge about possible consequences and their accurate evaluation.



(a) $\mathsf{st_{rA}}$ causes $\mathsf{loc_d}$

(b) $\mathsf{loc_d}$ requires $\mathsf{st_{rA}}$

(c) $\mathsf{st_{rA}}$ causes $\mathsf{loc_d}$ and $\mathsf{loc_d}$ requires $\mathsf{st_{rA}}$

Fig. 8. Risk graphs for $F = \{\mathsf{st_{rA}}, \mathsf{loc_d}\}$ after applying `causes` (a), `requires` (b), and both (c), using YAP

In summary, while $\leq_{\mathsf{m}}$ can deal with consequences shared by all risk factors (case 1), partially shared (case 2) or independent consequences (case 3) require a further investigation out of scope.

Condition 1 (Factor convergence) The use of $\leq_{\mathsf{m}}$ with several risk factors requires severity specifications to be based on a single compound consequence and a single severity scale across all factors.

The problematic case 3 could be modelled by an additional factor h that is caused by f *and* g and carries an up-shifted severity interval, for example, $\mathsf{h}.s = \mathsf{f}.s + \mathsf{g}.s$, and f, g, and h are all defined with respect to a single new consequence c_{A+B}. In Sect. 6.1 and Table 3, we have seen how such a dependency can be specified by constraints on the risk space, that is, by {f, g} `causes` h and h `excludes` {f, g}. `excludes` assures that severity calculation for states in phase h is only based on the interval $\mathsf{h}.s$.

Based on this analysis, we call a factor set $F \leq_{\mathsf{m}}$-*compatible* if the way of combining the severity intervals for each subset $F' \subseteq F$ fulfils Condition 1. For support in achieving and maintaining compatibility, the next paragraphs exemplify how factor severity and dependencies play together with mitigation orders. We also explore conditions for the well-formedness of severity intervals.

Factor characteristics and dependencies Constraints over risk spaces have implications on the characteristics of risk factors such as their severity intervals. Conversely, severity intervals govern $\leq_{\mathsf{m}}$ and can impose well-formedness conditions on the definition of constraints. In the following, we explore the impact of two different well-formedness conditions ($\mathsf{f}.s \subseteq \mathsf{g}.s$ and $\mathsf{f}.s \supseteq \mathsf{g}.s$) on $\leq_{\mathsf{m}}$ and the influence of choosing either one on the monotonicity of mitigation strategies. For example, for a constraint

$$\mathsf{f}\ \texttt{causes}\ \mathsf{g}, \quad \text{the condition} \quad \mathsf{f}.s \subseteq \mathsf{g}.s, \quad \text{and for} \tag{12}$$

$$F\ \texttt{causes}\ G, \quad \text{the condition} \quad \bigcup_{\mathsf{f}\in F} \mathsf{f}.s \subseteq \bigcup_{\mathsf{g}\in G} \mathsf{g}.s \tag{13}$$

seems coherent with the (probabilistic) relationship between the activation of causal and consequential factors. Particularly, the condition in Formula (12) expresses that if f causes g then f should have at most the potential impacts of g.

Consider the airbag spurious trip ($\mathsf{st_{rA}}$ with $s = [5, 12)$) causing loss of driving control ($\mathsf{loc_d}$ with $s = [2, 17)$) from the example in Table 2. For $\mathsf{st_{rA}}$ `causes` $\mathsf{loc_d}$, the two intervals would fulfil the aforementioned condition. The structure fulfilling this constraint is shown in Fig. 8a, it includes the states $\mathbf{0}$ with $S(\mathbf{0}) = [0, 0)$, loc_d with $S(loc_d) = [2, 17)$, and $loc_d\, st_{rA}$ with $S(loc_d\, st_{rA}) = [2, 17)$, with three equivalence classes in $R(\{\mathsf{st_{rA}}, \mathsf{loc_d}\})/_{=_{\mathsf{m}}}$ ordered according to

$$loc_d\overline{st_{rA}},\ loc_d\, st_{rA},\ loc_d \quad <_{\mathsf{m}} \quad \overline{loc_d}\, st_{rA},\ st_{rA} \quad <_{\mathsf{m}} \quad \overline{loc_d},\ \overline{st_{rA}},\ \mathbf{0},\ \overline{loc_d\, st_{rA}}\ .$$

On the contrary, assume a risk structure where $\mathsf{st_{rA}}.s = [2, 17)$ and $\mathsf{loc_d}.s = [5, 12)$ in order to be compliant with the converse condition $\mathsf{f}.s \supseteq \mathsf{g}.s$. From this structure, we obtain the order

$$\overline{loc_d}\,st_{rA},\ loc_d\,st_{rA},\ st_{rA} \quad <_{\mathsf{m}} \quad loc_d\overline{st_{rA}},\ loc_d \quad <_{\mathsf{m}} \quad \overline{loc_d},\ \overline{st_{rA}},\ \mathbf{0},\ \overline{loc_d}\,\overline{st_{rA}}\ .$$

In both orders, mitigation paths (i.e., sequences of solid arcs) in Fig. 8a, for example,

$$m^{\mathsf{loc_d}} \to m^{\mathsf{st_{rA}}} \to m_r^{\mathsf{st_{rA}}} \to m_r^{\mathsf{loc_d}} \to SKIP\ ,$$

remain in their equivalence class or lead to a state in a better class according to $\leq_{\mathsf{m}}$. Mitigations (solid arcs) deactivate at least one factor and, thus, lead to a better state according to $\prec_{\mathsf{m}}^{\sim}$. We call such paths through $\mathfrak{R}$ *mitigation monotonous* [GK17, Def. 8]. However, if intervals are specified in coherence with Formula (12), seven instead of five out of the twelve mitigations *strictly* reduce risk.

Analogously, for a constraint f `requires` g, the condition $\mathsf{f}.s \supseteq \mathsf{g}.s$ seems useful. The two factors in our example fulfil this condition for $\mathsf{loc_d}$ `requires` $\mathsf{st_{rA}}$. The structure fulfilling this constraint is shown in Fig. 8b, with the states ordered according to

$$loc_d\,st_{rA} \quad <_{\mathsf{m}} \quad \overline{loc_d}\,st_{rA},\ st_{rA} \quad <_{\mathsf{m}} \quad \overline{st_{rA}},\ \mathbf{0},\ \overline{loc_d}\,\overline{st_{rA}}\ ,$$

whereas intervals specified the opposite way (following $\mathsf{f}.s \subseteq \mathsf{g}.s$) would lead to a less distinctive order

$$\overline{loc_d}\,st_{rA},\ loc_d\,st_{rA},\ st_{rA} \quad <_{\mathsf{m}} \quad \overline{st_{rA}},\ \mathbf{0},\ \overline{loc_d}\,\overline{st_{rA}}\ .$$

Again, mitigation paths are monotonous and coherent with the above condition; four instead of three out of seven mitigations *strictly* reduce risk according to $\leq_{\mathsf{m}}$. Overall, a violation of the mentioned conditions does not influence mitigation monotonicity of this risk structure with respect to $\leq_{\mathsf{m}}$ but it reduces non-determinism in a strategy over $\mathfrak{R}$.

Figure 8c shows a risk graph for the discussed `causes/requires` constraint pair fulfilling both conditions. Note that the states are equivalent to Fig. 8b but the arc $\mathbf{0} \xrightarrow{e^{\mathsf{st_{rA}}}} st_{rA}$ in Fig. 8b is replaced by $\mathbf{0} \xrightarrow{e^{\mathsf{loc_d}\,\mathsf{st_{rA}}}} st_{rA}\,loc_d$ and the edge $st_{rA}\,loc_d \xrightarrow{m_s^{\mathsf{st_{rA}}}.m_s^{\mathsf{loc_d}}} \overline{st_{rA}}\,\overline{loc_d}$ is added. In summary, the `causes` constraint streamlines the transition relation of the risk structure as opposed to the `requires` constraint. This can be useful for merging factors during reduction of a risk structure. However, Fig. 8a shows that `causes` is less restrictive than `requires`, particularly, allowing $e^{\mathsf{loc_d}}$ to occur without a previous $e^{\mathsf{st_{rA}}}$, for example, due to other causes not made explicit as factors in the model.

An in-depth analysis of techniques for making F compatible, a discussion of a set of rules relating factor characteristics and factor dependencies, as well as an alternative to $\leq_{\mathsf{m}}$ are subject of future work.

7. RiskStructures: the language

In the previous sections, we have explored how a risk space R can be used for assessing *risk mitigation capabilities* of a process P (cf. Fig. 1) and for equipping P with a form of *risk awareness*. Risk in specific situations has a *specific causal structure*. We have seen how *dependencies* in a factor set F focus on this structure. Starting from the full extension of R, we *select subsets* of R, making assumptions if the structure is not entirely known in advance. This focus and these assumptions can be expressed by a constrained *composition*. Specifically, based on *constraints* on the combinations of factor phases, on the phase transitions, and on the *synchronisation* of events corresponding to the CSP model of concurrency, we can specify

- which region of R we want to pay attention to (i.e., the scope of safety guarantees) and
- which region of R we consider to be safe for a process P (i.e., conventional safety).

We use LTSs for investigating the operational semantics of RiskStructures, beginning with factors and their constrained composition, and accompanied by a discussion of the *consistency, well-formedness, and validity* of their algebraic semantics.

Definition 14 (Risk structure) Given a factor set $F \subseteq \mathcal{F}$, a *risk structure* $\mathfrak{R}$ is an expression of the form

$$\mathfrak{R} ::= p \mid \mathfrak{R} \parallel \mathfrak{R} \mid [\mathfrak{R}]_C$$

where, for a factor $\mathsf{f} \in F$ (Definition 3), phase $p \in Ph_{\mathsf{f}}$ (e.g. $0^f, f, \overline{f}$) is a risk structure and $C \subseteq \mathcal{C}$ is a set of *constraints* (Definition 12). Let $\mathcal{S}$ be the set of all risk structures, including all phases $\bigcup_{\mathsf{f}\in\mathcal{F}} Ph_{\mathsf{f}} \subset \mathcal{S}$.

The binary operator $\|$ signifies the parallel composition of two structures, and the binary operator $[\cdot]_C$ applies all constraints in the set C to a risk structure.

Operational semantics Let the map $[\![\cdot]\!]_\mathsf{r} : \mathcal{S} \to \mathcal{T}$ uniquely assign a LTS to a CSP process corresponding to a risk structure. With $\mathfrak{R} \in \mathcal{S}$ we associate a LTS $[\![\mathfrak{R}]\!]_\mathsf{r} = (R, \Sigma^u, \to, R_0)$ with the risk space R (Definition 5), the abstracted used alphabet $\Sigma^u \subset 2^{\Sigma^\tau} \setminus \{\varnothing\}$ (Definition 1, Sects. 7.1, 7.2, 7.3), the transition relation $\to \subseteq R \times \Sigma^u \times R$ (Definitions 15 and 16, Sect. 7.3), and a set of initial states $R_0 \subseteq R$. The operational semantics and algebraic properties of the operators and constructs of the language in Definition 14 are provided below. The construction of the used alphabet Σ^u will be explained along with the operators.

7.1. Risk factors as sequential processes

A risk factor $\mathsf{f} \in F$ (Figs. 2 and 3) can be represented as a sequential, mutually recursive process $\mathfrak{R}_\mathsf{f}$ according to Definition 1. We express f as

$$\mathfrak{R}_\mathsf{f} = 0^f \qquad \text{(init)}$$
$$0^f = ?x : e^\mathsf{f} \to f \,\Box\, ?x : o^\mathsf{f}_n \to 0^f \qquad \text{(inactive)}$$
$$f = ?x : m^\mathsf{f}_d \to 0^f \,\Box\, ?x : o^\mathsf{f}_e \to f \,\Box\, ?x : m^\mathsf{f} \to \overline{f} \qquad \text{(active)}$$
$$\overline{f} = ?x : \overline{e}^\mathsf{f} \to f \,\Box\, ?x : m^\mathsf{f}_r \to 0^f \,\Box\, ?x : o^\mathsf{f}_m \to \overline{f}. \qquad \text{(mitigated)}$$

Note the use of f as a symbol for the risk factor as a transition system (Fig. 2a) and the use of f for the CSP process that models this factor in its active phase. Different fonts signify the semantic difference. Also notice that $p \xrightarrow{\tau}$ for any $p \in Ph_\mathsf{f}$ of a non-deterministic factor (Fig. 4) corresponds to the $\sqcap$-fraction in $\Box$.
This mapping enables an algebraic treatment of factors in CSP.

The semantics of a single factor f in phase $p \in Ph_\mathsf{f}$ is given by $[\![p]\!]_\mathsf{r} = (R(\{\mathsf{f}\}), \Sigma^\mathsf{f}, \to_\mathsf{f}, \{[\mathsf{f} \mapsto p]\})$ and the elements of this tuple are given by Definition 3 and described in Fig. 2. For example, $[\![0^f]\!]_\mathsf{r} = (R(\{\mathsf{f}\}), \Sigma^\mathsf{f}, \to_\mathsf{f}, \{[\mathsf{f} \mapsto 0^f]\})$ provides the semantics of f initialised with the phase 0^f. For each factor, we assume a phase order $\preceq_\mathsf{f}$ as given in Sect. 4.1.

Factor alphabets for global input-enabledness Based on the definitions in Sect. 4.1, the condition

$$\forall\, \mathsf{f}, \mathsf{g} \in F : \Big(\bigcup_{e \in \Sigma^\mathsf{f}} e\Big) \setminus \{\tau\} = \Big(\bigcup_{e' \in \Sigma^\mathsf{g}} e'\Big) \setminus \{\tau\} = \Sigma \tag{14}$$

requires all factors to support the alphabet Σ of the process P. This choice does not affect risk space composition (Sect. 4.2) but allows factors to be composed in parallel synchronously and be ready to agree with some event P offers. This attempt to avoid interleaving requires input-enabledness of $\mathfrak{R}$ (Sect. 4.1). Given F in terms of a factor family $(\mathsf{f}_i)_{i \in [1..n]}$ with $n \in \mathbb{N}$, one way to satisfy Formula (14) is a structured alphabet with compound (i.e., multi-part) events

$$r.\alpha_1.\alpha_2 \ldots \alpha_n$$

with $\alpha_1.\alpha_2 \ldots \alpha_n \in \Sigma^{\mathsf{f}_1}.\Sigma^{\mathsf{f}_2} \ldots \Sigma^{\mathsf{f}_n}$ and a central channel r of type $\Sigma^{\mathsf{f}_1}.\Sigma^{\mathsf{f}_2} \ldots \Sigma^{\mathsf{f}_n}$ through which $\mathfrak{R}$ can observe and influence P. For $i, j \in [1..n]$ with $i \neq j$, two (partially defined) factor event sets, say an endangerment $e^{\mathsf{f}_i} \in \Sigma^{\mathsf{f}_i}$ activating factor f_i and a mitigation $m^{\mathsf{f}_j} \in \Sigma^{\mathsf{f}_j}$ of factor f_j, then correspond to the set of (fully defined) compound events given by

$$events_\mathfrak{R}(r?e^{\mathsf{f}_i}!m^{\mathsf{f}_j}) = \{r.\alpha_1 \ldots ?e^{\mathsf{f}_i} \ldots !m^{\mathsf{f}_j} \ldots \alpha_n \mid \alpha_1 \in \Sigma^{\mathsf{f}_1}, \ldots, e^{\mathsf{f}_i} \in \Sigma^{\mathsf{f}_i}, \ldots, m^{\mathsf{f}_j} \in \Sigma^{\mathsf{f}_j}, \ldots, \alpha_n \in \Sigma^{\mathsf{f}_n}\}\,.$$

At the level of P, these compound events are equivalently given by

$$events_P(r?e^{\mathsf{f}_i}!m^{\mathsf{f}_j}) = \{r.x_1 \ldots x_i \ldots x_j \ldots x_n \mid x_1 \in \bigcup_{e \in \Sigma^{\mathsf{f}_1}} e, \ldots, x_i \in e^{\mathsf{f}_i}, \ldots, x_j \in m^{\mathsf{f}_j}, \ldots, x_n \in \bigcup_{e \in \Sigma^{\mathsf{f}_n}} e\}\,.$$

This channel type is inspired by suggestions in [Ros10, Sec. 1.2.4] and by [BS01, Sec. 3.4]. Thus, each factor f synchronises with the projection from Σ into Σ^f and is always ready to agree with some event offered by

P.[21] This construction models partial observability of P through factors implemented by sensors and actuators and, for a deterministic factor set F, guarantees deadlock freedom of $P \parallel \mathfrak{R}$, modelling that a corresponding implementation of F can always monitor and influence P.

7.2. Composition

Let $A, B, F \subseteq \mathcal{F}$ be factor sets with $A, B \subseteq F$ and, for $i \in \{1,2,3\}$, $\mathfrak{R}_i \in \mathcal{S}$ be structures according to Definition 14 with $[\![\mathfrak{R}_i]\!]_\mathsf{r} = (R_i, \Sigma_i^u, \to_i, R_{i,0})$. Now, we may want to combine two structures, say $\mathfrak{R}_1$ and $\mathfrak{R}_2$, with possibly intersecting factor sets, for example, if two safety engineers are trusted with two HazOps of the same machine. Let the map $scope\colon \mathcal{S} \to 2^{\mathcal{F}}$ identify the factors referred to by a structure. Then, $scope(\mathfrak{R}_1) \cap scope(\mathfrak{R}_2)$ indicates all shared factors.

Because a single factor cannot be in two different phases at the same time, we apply Definition 6 for $F_1 = scope(\mathfrak{R}_1)$ and $F_2 = scope(\mathfrak{R}_2)$ to uniquely define the transition relation resulting from parallel composition: *only those states can be combined whose factors in the shared scope are in their identical phases.*

Definition 15 (Parallel composition) We define $[\![\mathfrak{R}_1 \parallel \mathfrak{R}_2]\!]_\mathsf{r} = (R, \Sigma^u, \to, R_0)$ with $R = R_1 \otimes R_2$ (see Definition 7 and Lemma 1) and $R_0 = R_{1,0} \otimes R_{2,0}$. For states $\sigma_1, \sigma_1' \in R_1, \sigma_2, \sigma_2' \in R_2$, the composed transition relation $\to$ is given by the *left, right, and synchronous* step rules

$$\frac{\sigma_1 \xrightarrow{e}_1 \sigma_1' \qquad \sigma_2 \xrightarrow{f}_2 \sigma_2'}{\sigma_1 \cup \sigma_2 \xrightarrow{e \setminus f} \sigma_1' \cup \sigma_2} \; [\sigma_1 \approx \sigma_2, \sigma_1' \approx \sigma_2, e \neq f] \;, \tag{$\|$-l-step}$$

$$\frac{\sigma_1 \xrightarrow{e}_1 \sigma_1' \qquad \sigma_2 \xrightarrow{f}_2 \sigma_2'}{\sigma_1 \cup \sigma_2 \xrightarrow{f \setminus e} \sigma_1 \cup \sigma_2'} \; [\sigma_1 \approx \sigma_2, \sigma_1 \approx \sigma_2', e \neq f] \;, \text{ and} \tag{$\|$-r-step}$$

$$\frac{\sigma_1 \xrightarrow{e}_1 \sigma_1' \qquad \sigma_2 \xrightarrow{f}_2 \sigma_2'}{\sigma_1 \cup \sigma_2 \xrightarrow{e \cap f} \sigma_1' \cup \sigma_2'} \; [\sigma_1 \approx \sigma_2, \sigma_1' \approx \sigma_2', e \cap f \neq \varnothing] \;. \tag{$\|$-s-step}$$

Based on $\to$, the *used alphabet* is $\Sigma^u = \{e \in 2^{\Sigma^\mathrm{r}} \mid \exists \sigma, \sigma' \in R\colon \sigma \xrightarrow{e} \sigma'\}$.

Note that $\|$-l-step covers the special case $f = \varnothing$ corresponding to $\sigma_2 \nrightarrow_2$. $\|$-r-step works analogously. These step rules together resemble the step law of generalised parallel composition in CSP [Ros10, Sec. 3.4].

Algebraic properties of composition In the following, we show some desirable properties of the composition of two or more risk structures.

Lemma 8 (Idempotency of $\|$ under determinism) For any *deterministic* $\mathfrak{R}_1 \in \mathcal{S}$, we have

$$\mathfrak{R}_1 \parallel \mathfrak{R}_1 = \mathfrak{R}_1 \;. \tag{$\|$-idem}$$

Proof of Lemma 8. From Definition 5, by Lemma 1, we obtain $R_1 \otimes R_1 = R_1$. Based on Sect. 4.2 and the alphabet of $\mathfrak{R}_\mathsf{f}$ defined in Sect. 7.1, we can apply the rules from Definition 15 as follows.

Because of the uniqueness of $\mathfrak{R}_1$ and the compatibility requirement (Formula (2)), the composition takes two consistent views of $\mathfrak{R}_1$ offering identical source and target states ($\sigma_1 = \sigma_2$) at each step. By definition, all factors share the alphabet Σ and $\mathfrak{R}_1$ is deterministic. Then, any two event sets e, f in $initials(\mathfrak{R}_1)$ (i.e., in a state σ) are either disjoint or equal. Hence, there is no way for a concrete event in P to be a member of more than one such event set. In line with Definition 3, $\|$-l-step and $\|$-r-step do not apply. $\|$-s-step turns into a tautology and both views exhibit an identical transition:

$$\frac{\sigma_1 \xrightarrow{e}_1 \sigma_1'}{\sigma_1 \cup \sigma_1 \xrightarrow{e} \sigma_1' \cup \sigma_1'} \;.$$

[21] According to the CSP notation for channel events, we have $\Sigma = \{\!|\, r \,|\!\} = events(r)$ for the channel r.

For $\sigma_1, \sigma_1' \in R_1$ and $e \in \Sigma_1^u$, this tautology preserves Σ_1^u, $\to_1$, and $R_{1,0}$. □
The composition of a deterministic $\mathfrak{R}_1$ with itself turns into a synchronous (i.e., non-interleaved) composition, essentially, a sequence of external choices where both $\mathfrak{R}_1$s synchronously agree with the environment.

Lemma 9 (Commutativity of ‖) For any $\mathfrak{R}_1, \mathfrak{R}_2 \in \mathcal{S}$, we have

$$\mathfrak{R}_1 \parallel \mathfrak{R}_2 = \mathfrak{R}_2 \parallel \mathfrak{R}_1 \,. \tag{‖-comm}$$

Proof of Lemma 9. Lemma 1 in Sect. 4.2 makes it easy to show $R_1 \otimes R_2 = R(scope(\mathfrak{R}_1) \cup scope(\mathfrak{R}_2)) = R_2 \otimes R_1$. Furthermore, the symmetric duals (i.e., ‖-r-step is the dual of ‖-l-step) of the rules in Definition 15 yield the same $\to$ and, hence, the same Σ^u. □

Lemma 10 (Associativity of ‖) For any $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3 \in \mathcal{S}$ with *disjoint* scopes, that is,

$$scope(\mathfrak{R}_1) \cap scope(\mathfrak{R}_2) \cap scope(\mathfrak{R}_3) = \varnothing,$$

we have

$$(\mathfrak{R}_1 \parallel \mathfrak{R}_2) \parallel \mathfrak{R}_3 = \mathfrak{R}_1 \parallel (\mathfrak{R}_2 \parallel \mathfrak{R}_3) \,. \tag{‖-assoc-1}$$

Proof sketch for Lemma 10. Because of empty shared scopes, the side conditions always hold on both sides and states are merged by disjoint union. Set operations on states and events are commutative and associative. By definition,[22] all factors (Definition 3) share the alphabet Σ, that is, they synchronise on all events (Sect. 4.2). Thus, interleaving is avoided, the alphabetised parallel operator takes Σ on both sides and, this way, reduces to synchronous parallel composition for which a general associative law is available [Ros10, p. 60]. □
Lemma 10 allows the use of $\|^{\mathsf{f} \in F}\ 0^f$ as a shortcut for $((\ldots(0^{f_1} \parallel 0^{f_2}) \parallel \ldots) \parallel 0^{f_n})$ with $\mathsf{f}_i \in F$.

Lemma 11 (Associativity of ‖) For any $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3 \in \mathcal{S}$ with *equal* scopes, that is,

$$scope(\mathfrak{R}_1) = scope(\mathfrak{R}_2) = scope(\mathfrak{R}_3) \,,$$

we have

$$(\mathfrak{R}_1 \parallel \mathfrak{R}_2) \parallel \mathfrak{R}_3 = \mathfrak{R}_1 \parallel (\mathfrak{R}_2 \parallel \mathfrak{R}_3) \,. \tag{‖-assoc-2}$$

Proof sketch of Lemma 11. The proof is analogous to the proof of Lemma 10. □

7.3. Constraints

In Sect. 6, we specified relations over risk spaces to determine $\to$ by pruning $R \times R$. We now use constraints as a construct in the language of RISKSTRUCTURES (Definition 14). The redundancy coming with this construct can be used to identify inconsistencies between $\mathfrak{R}$ and the real world, potentially helpful in *model refinement*, *completion*, and *validation* [Gle14]. Particularly, these inconsistencies allow choices for their resolution. We can make such inconsistencies explicit as follows.

For factors $F \subseteq \mathcal{F}$, constraints $C \subseteq \mathcal{C}$, and a risk state $\sigma \in R(F)$, we call

$$\mathfrak{R}_\sigma^C = [\|^{\mathsf{f} \in F}\ \sigma(\mathsf{f})]_C$$

with $R_0 = \{\sigma\}$ the *characteristic risk structure* of σ under constraints C. *But which definition of $\sigma \to$ is most useful?* To investigate this, we split the determination of $\sigma \to$ for $\mathfrak{R}_\sigma^C$ into three fractions $\sigma \to_{\mathsf{c,f,p}}$ with

$\sigma \to_{\mathsf{c}}$	derived from C, with	$\sigma \xrightarrow{\tau_{\mathsf{c}}} \sigma' \in \sigma \to_{\mathsf{c}} \iff (\sigma, \sigma') \in [\![C]\!]_{\mathsf{c}}^R$,
$\sigma \to_{\mathsf{f}}$	from Definitions 3 and 15, with	$\sigma \xrightarrow{e} \sigma' \in \sigma \to_{\mathsf{f}} \iff \sigma \xrightarrow{e} \sigma'$, and
$\sigma \to_{\mathsf{p}}$	from process P, with	$\sigma \xrightarrow{e} \sigma' \in \sigma \to_{\mathsf{p}} \iff e \in initials(P) \wedge \sigma' \in reach(\sigma, P)$.

These fractions give rise to the following *inconsistencies*:

1. The relation $\sigma \to_{\mathsf{c}} \setminus \sigma \to_{\mathsf{f}}$ describes *sensible* transitions with *invisible events* signified by τ_{c}. We choose to prune transitions from $\to$ if they deviate from what is provided in $\to_{\mathsf{f}}$.

[22] We provide the rationale for this definition in Sect. 7.5.

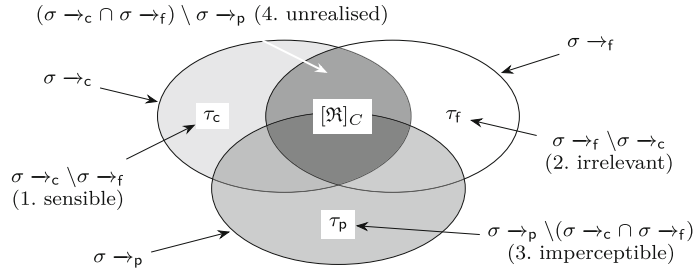


Fig. 9. Semantics of $\sigma \to$ for the constrained form $[\mathfrak{R}]_C$ according to Definition 16

2. The relation $\sigma \to_{\mathsf{f}} \setminus \sigma \to_{\mathsf{c}}$ describes *irrelevant* transitions labelled with a τ_{f}. We choose to prune transitions from $\to$ if they violate constraints and, hence, lead to inconsistencies in $\mathfrak{R}$.
3. The relation $\sigma \to_{\mathsf{p}} \setminus (\sigma \to_{\mathsf{c}} \cup \sigma \to_{\mathsf{f}})$ describes *imperceptible* transitions. In $\to$, we could label such transitions with a τ_{p}, making them subject of *process-driven disclosure* of $\mathfrak{R}$.
4. The relation $(\sigma \to_{\mathsf{c}} \cap\, \sigma \to_{\mathsf{f}}) \setminus \sigma \to_{\mathsf{p}}$ describes *unrealised* transitions. We could prune such transitions from $\to$, because they are not realised in P and, hence, would only add little value to $\mathfrak{R}$.

This case analysis is visualised in Fig. 9 and suggests several possibilities to design a semantics for constraints. As indicated in the case list, one would desire the following possibility for $\to$:

$$\begin{aligned}\sigma \to = (\sigma \to_{\mathsf{c}} \cup \sigma \to_{\mathsf{f}} \cup \sigma \to_{\mathsf{p}}) \setminus \big(&\underbrace{(\sigma \to_{\mathsf{c}} \setminus \sigma \to_{\mathsf{f}})}_{\text{1. reduce to } F} \cup \underbrace{(\sigma \to_{\mathsf{f}} \setminus \sigma \to_{\mathsf{c}})}_{\text{2. reduce to } C} \cup \underbrace{(\sigma \to_{\mathsf{c}} \cup \sigma \to_{\mathsf{f}} \setminus \sigma \to_{\mathsf{p}})}_{\text{4. reduce to } P} \big) \\ = \sigma \to_{\mathsf{c}} \cap\, \sigma \to_{\mathsf{f}} \cap\, \sigma \to_{\mathsf{p}} \ . &\qquad \text{(by set-algebraic manipulation)}\end{aligned}$$

Corollary 6 For any characteristic structure $\mathfrak{R}^C_\sigma$, we have $\sigma \to_{\mathsf{c}} \setminus \sigma \to_{\mathsf{f}} = \varnothing$ because of the input-enabledness [Tre08] of factors and, thus, $\sigma \to = \sigma \to_{\mathsf{c}} \cap\, \sigma \to_{\mathsf{p}}$.

To weaken the dependency of $\mathfrak{R}$ on a possibly partially known process, we apply the following definition.

Definition 16 (Constraint) Let $\mathfrak{R}$ be a risk structure (Definition 14) with $[\![\mathfrak{R}]\!]_{\mathsf{r}} = (R, \Sigma^u, \to, R_0)$ and C a set of constraints well-formed for R (Sect. 6, Definition 13). The *constrained form* $[\mathfrak{R}]_C$ is defined by $[\![[\mathfrak{R}]_C]\!]_{\mathsf{r}} = (R, \Sigma^u_C, \to_C, R_0)$. $\to_C$ is determined by the following two rules. For $\sigma, \sigma' \in R$

$$\frac{\sigma \xrightarrow{e} \sigma' \qquad (\sigma, \sigma') \in [\![C]\!]^R_{\mathsf{c}}}{\sigma \xrightarrow{e}_C \sigma'} \quad \text{and} \qquad ([\cdot]_C\text{-step})$$

$$\frac{\sigma \xrightarrow{e} \sigma' \qquad (\sigma, \sigma') \notin [\![C]\!]^R_{\mathsf{c}}}{\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma} \ . \qquad ([\cdot]_C\text{-cancel})$$

Based on $\to_C$, $\Sigma^u_C = \{e \in 2^{\Sigma^{\mathsf{r}}} \mid \exists \sigma, \sigma' \in R \colon \sigma \xrightarrow{e}_C \sigma'\}$.

For arbitrary structures, the $[\cdot]_C$-step rule implements $\to = \to_{\mathsf{c}} \cap \to_{\mathsf{f}}$ (i.e., cases 1 and 2). The reduction of unrealised transitions and the coverage of imperceptible transitions (i.e., cases 3 and 4) could be addressed by an incremental construction of $\mathfrak{R}$. $[\cdot]_C$-cancel produces τ_{f} self-loops to preserve input-enabledness.[23] Notice that τ_{f}-loops remain whether or not they are contained in further constraints applied to $\mathfrak{R}$, and also that Corollary 6 and Definition 16 prevent constraints such as causes from producing τ_{c}. Overall, the rules $\|$-l-step, $\|$-r-step, $\|$-s-step, $[\cdot]_C$-step, and $[\cdot]_C$-cancel determine the operational semantics of a risk structure as an LTS and, together with *reach* (Sect. 5.4), form a basis for automated reasoning about RiskStructures. We demonstrate such automation with our tool prototype Yap [Gle21].

Remark 2 The meaning of f causes g, specified as a relation in Sect. 6.1, can now be investigated in the LTS semantics of $\mathfrak{R}$. On a particular *path* through R according to $\to_C$ from a state in R_0, whenever f gets active, g must

[23] Operationally, $[\cdot]_C$-cancel acts like $SKIP$, non-blocking $STOP$, or a hiding of trace suffixes. Hence, we define $\tau_{\mathsf{f}} = \checkmark$.

get active in the same or the following state. This way, `causes` constraints allow *immediate or weak causation* and `requires` constraints *delayed or strong causation* at most one logical step apart in $\mathfrak{R}$. The weak form of `causes` enforces the *simultaneous* occurrence of `f` and `g`, it *synchronises* the activation of `g` with the prior activation of `f`. Moreover, `g` can stay active forever and may already be active before the activation of `f`. Consider that `g` might have been activated by a factor or a causal relationship not (yet) captured in $\mathfrak{R}$. Such activation is by default possible in $\mathfrak{R}$ once `g` is in the factor set.

Algebraic properties of constraints ($[\cdot]_C$). Let $C_i \subseteq \mathcal{C}$ for $i \in \{1, 2, 3\}$. We have that $[\mathfrak{R}]_\varnothing = \mathfrak{R}$ by Definition 12. The following lemma establishes that the order in which single constraints are applied to $\mathfrak{R}$ does not matter.

Lemma 12 (Exchange of $[\cdot]_C$ and $\cup$)

$$[[\mathfrak{R}]_{C_1}]_{C_2} = [\mathfrak{R}]_{C_1 \cup C_2} \tag{15}$$

Proof sketch. The proof is by induction over C_1, supported by an additional lemma for the induction step, and takes advantage of associativity of set intersection as used in Definition 12. The detailed proof is stated in Appendix A. □

Lemma 13 (Idempotency)

$$[[\mathfrak{R}]_C]_C = [\mathfrak{R}]_C \tag{$[\cdot]_C$-idem}$$

Proof sketch. This lemma follows from Lemma 12 and idempotency of set union. □

Lemma 14 (Commutativity)

$$[[\mathfrak{R}]_{C_1}]_{C_2} = [[\mathfrak{R}]_{C_2}]_{C_1} \tag{$[\cdot]_C$-comm}$$

Proof sketch. This lemma follows from Lemma 12 and commutativity of set union. □

Lemma 15 (Associativity)

$$[[\mathfrak{R}]_{C_1 \cup C_2}]_{C_3} = [[\mathfrak{R}]_{C_1}]_{C_2 \cup C_3} \tag{$[\cdot]_C$-assoc}$$

Proof sketch. This lemma follows from Lemma 12 and associativity of set union. □

For an arbitrary $C \subseteq \mathcal{C}$, we relax Definition 16 by establishing following equivalence

$$[\mathfrak{R}]_C = [\mathfrak{R}]_{C'} \tag{16}$$

for the largest $C' \subseteq C$ well-formed for R (Definition 13). Then, $[\mathfrak{R}]_C$ denotes the structure resulting from applying only and exactly the constraints in C'. Moreover, Definition 16 complements Definition 15 by guarding the $\|$-l-step, $\|$-r-step, and $\|$-s-step rules. In this way, the application of a constraint $c \in \mathcal{C}$ to a risk structure $\mathfrak{R}$ restricts its transition relation $\rightarrow$. Definition 16, together with the relational pruning for three out of the four mentioned cases (cf. Fig. 9), yields the following refinement law.

Lemma 16 (Constraints trace-refine)

$$\mathfrak{R} \sqsubseteq_T [\mathfrak{R}]_C$$

Proof sketch. The proof is by showing that $traces(\mathfrak{R}) \supseteq traces([\mathfrak{R}]_C)$. Fix $t \in traces([\mathfrak{R}]_C)$ with $t = f^\frown l$ for induction over the split of t into f and l. Induction step: Assume that $f \in traces(\mathfrak{R})$ as *induction hypothesis* (IH). With $l = \langle e \rangle^\frown l'$ and $t = f^\frown\langle e \rangle^\frown l'$, there exist $\sigma, \sigma' \in R$ such that $\sigma \xrightarrow{e}_C \sigma'$ and, according to *rule* $[\cdot]_C$-step, such that $e \in \Sigma^u$, $\sigma \xrightarrow{e} \sigma'$, and $(\sigma, \sigma') \in [\![C]\!]^R_{\mathsf{c}}$. Because of IH and $\sigma \xrightarrow{e} \sigma'$, there must be a trace $f^\frown\langle e \rangle^\frown l'' \in traces(\mathfrak{R})$ and, because of downward closure, a trace $f^\frown\langle e \rangle$. e is part of l and, therefore, t such that we complete the induction step and establish the new IH $f^\frown\langle e \rangle$. (We do not need to prove the equivalence $l' = l''$.) The case $f = \langle\rangle$ provides the induction start and $l = \langle e \rangle$ terminates the induction. □

Following the pruning semantics described in Fig. 9, Lemma 16 establishes the desirable effect of constraints, the pruning of the transition relation of $\mathfrak{R}$ according to the relational specification of the added constraints. However, an investigation of the conditions under which constraints are compositional, that is, when they can be exchanged with, for example, $\|$, is left for future work.

Listing 2: YAP script modelling the situation `manual` from Table 2

```
factormodel for manual {
  // single compound consequence on a single asset: impact on human driver
  col desc "collision"
    accident
    detectedBy(.crashSens)
    mitigatedBy(.rA)
    requires(ncol)
    sev=[10,20);

  strA desc "airbag spurious trip"
    causes(locd)
    sev=[5,12);
  locd desc "loss of driving control"
    preventsMit(ncol,locd) // mitigation lock at loc
    requires(strA)
    sev=[2,17);
  ncol desc "near collision"
    detectedBy(.ODM)
    mitigatedBy(PREVENT_CRASH.swBr)
    direct
    sev=[5,18);
}

controlloop vehOnRd {
  mode ODM desc "object detection and distance measurement system"
    guard "distToNOoT < SBD" // distance To Nearest Object on Trajectory Is Less Than Safe Braking Distance
    embodiedBy lidar;

  mode SwBr desc "swerve and brake"
    type PREVENT_CRASH
    target (act=SwBr); // perform low-level procedure

  mode crashSens desc "crash sensor"
    update "shock=true"; // forward validated shock signal to airbag

  mode rA desc "release driver and passenger airbags"
    guard "shock=true"
    update "release=DrvAndPsgr"
    embodiedBy airbag;

  lidar;
  airbag;
}
```

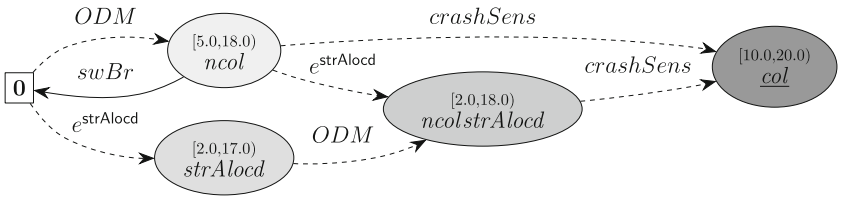


Fig. 10. Risk structure $\mathfrak{R}_{\mathsf{manual}}$ with the factor set $F = \{\mathsf{loc_d}, \mathsf{ncol}, \mathsf{st_{rA}}, \mathsf{col}\}$ generated by YAP from Listing 2

Finally, we discuss the special case that constraints are applied to a phase $p \in Ph_{\mathsf{f}}$ of a factor f. Observe that $scope(p) = \{\mathsf{f}\}$ and, by Definition 13, C is well-formed for $R(scope(p))$ if C contains only constraints that refer to f. Moreover, if C does not contain self-referential constraints (Sect. 6.1), Formula (16) entails $C' = \varnothing$ for the largest well-formed $C' \subseteq C$. From this observation, we obtain

$$[p]_C \underbrace{= [p]_{C'}}_{\text{by Formula (16)}} \underbrace{= [p]_{\varnothing}}_{\text{by Definition 13}} \underbrace{= p}_{\text{by Definition 12}} .$$

The effect of this last result is that single factors are not affected by constraints other than self-referential ones. Self-referential constraints such as f `direct` can, however, restrict p or convert p to $SKIP$. Often, such constraints can be avoided because they do not offer much expressiveness over bespoke factor models.

7.4. Example: risk factors on the road (cont'd)

We develop the example in Sect. 3 and sects. 4.3 and 6.3, modelling instances of the factor set $F = \{\mathsf{loc_d}, \mathsf{ncol}, \mathsf{st_{rA}}, \mathsf{col}\}$ with corresponding events and compose these factors into the structure

$$\mathfrak{R}_{\mathsf{manual}} = [((0^{st_{rA}} \parallel 0^{loc_d}) \parallel 0^{ncol}) \parallel 0^{col}]_C$$

for the situation `manual`. C contains dependencies such as $\mathsf{loc_d}$ `preventsMit` $\{\mathsf{ncol}, \mathsf{loc_d}\}$ and $\mathsf{st_{rA}}$ `causes` $\mathsf{loc_d}$. The full dependency list can be derived from Listing 2. Figure 10 shows the transition graph for $\mathfrak{R}_{\mathsf{manual}}$ after applying composition and constraints. The edges are labelled with factor events and the nodes with active factors. Listing 2 is written in YAP script, the input language for our tool YAP [Gle21], which allows one to encode a situation/action/factor table (Table 1a) and elaborate the example in Sect. 3 in several steps.

Collision (col) is the first phenomenon we model as a factor. With the `detectedBy` directive, we specify that a collision is detected by a crash sensor ($crashSens$) that signals the release of an airbag (rA). We associate an interval $\mathsf{col}.s = [10, 20)$ of 10 to 20 "negativity units" to a collision.

To illustrate *scoping* of a risk structure, we use the keyword `accident` to declare col as final. Mitigation of final factors is *out of scope*, so vehicle control would not take into account $crashSens$ and rA in this case. However, in practice, col is not final because of being mitigated by an airbag. Capturing a step of accident analysis–technically, backwards causal reasoning–the dependency col `requires` ncol specifies that a near-collision is assumed to occur before any collision.

A *near-collision* (ncol) is detected by an object distance measurement system (`ODM`). The `ODM` is responsible for monitoring the predicate $distToNOoT < SBD$, which is true if the distance measured to the nearest object on the road within the planned vehicle trajectory ($distToNOoT$) is smaller than the safe braking distance SBD. We assume an interval $\mathsf{ncol}.s = [5, 18)$ with a smaller upper bound than the one for col because the indirection to the accident reduces the possibility of bad consequences. Concretely, the likelihood of consequences from col is reduced by the action $swBr$. Based on the discussion in Sect. 4.4, we plan in future work to quantify this likelihood through probabilistic factor transitions.

The `mitigatedBy(`*`PREVENT_CRASH.swBr`*`)` directive specifies as a mitigation of ncol the *swerve-and-break* manoeuvre ($swBr$). The constraint `direct` specifies that $swBr$ to be the event m_d^{ncol} (cf. Fig. 2a). Shown in the `controlloop` block, $swBr$ is a low-level control `mode` to be elaborated in an external model (e.g. in MATLAB or its free alternative GNU/Octave[24]). *`PREVENT_CRASH`* is a classifier from which generic parameters for low-level controllers of this kind could be inherited. The modelling of modes is akin to the specification of guarded commands (i.e., using the `guard` and `update` attributes). We demonstrate in [GC20] how YAP modes can be used for controller synthesis, and in [FGC20] how factor LTSs can be modelled as hybrid automata, simulated in GNU/Octave, and formally verified in Isabelle/UTP.

By FMEA of the airbag, we identify its *spurious trip* ($\mathsf{st_{rA}}$) and assume it to be *one of possibly several sufficient causes* of a *loss of driving control* ($\mathsf{loc_d}$). This finding is captured by the two dependencies $\mathsf{st_{rA}}$ `causes` $\mathsf{loc_d}$ and $\mathsf{loc_d}$ `requires` $\mathsf{st_{rA}}$. $\mathsf{st_{rA}}$ gets assigned the interval $\mathsf{st_{rA}}.s = [5, 12)$.[25] We specify the interval $\mathsf{loc_d}.s = [2, 17)$. Moreover, $\mathsf{loc_d}$ `preventsMit` {ncol, $\mathsf{loc_d}$} captures the fact that after a loss of control, `manual` driving mode prevents the mitigation of ncol. Additionally, $\mathsf{loc_d}$ locks itself using a circularity so that, in $\mathfrak{R}_{\mathsf{manual}}$, it cannot be mitigated.

From the severity intervals calculated for each risk state according to Formula (5) (see the state labels in Fig. 10), we establish

$$\underline{col} <_{\mathsf{m}} ncol =_{\mathsf{m}} loc_d\, st_{rA}\, ncol <_{\mathsf{m}} loc_d\, st_{rA} <_{\mathsf{m}} \mathbf{0}\ .$$

Notice that, according to $\leq_{\mathsf{m}}$, the state $ncol$ is equivalent to $loc_d\, st_{rA}\, ncol$. This equivalence results from the equality of the upper bounds of both intervals [5, 18) and [2, 18). Although one might think that the state with the smaller lower bound (i.e., $loc_d\, st_{rA}\, ncol$) is better, consider that $ncol$ has two active factors less. This circumstance gives rise to investigate refinements of $\leq_{\mathsf{m}}$. In fact, our tool YAP [Gle21] implements a special case of $\leq_{\mathsf{m}}$ distinguishing pairs (σ, σ') of risk states that normally are $\sigma =_{\mathsf{m}} \sigma'$ because they only differ in the lower bounds of their severity intervals.

The state $\mathbf{F}$ with all factors activated (cf. Definition 11) is not reachable from $\mathbf{0}$. Moreover, only one of the five relevant states in $\mathfrak{R}_{\mathsf{manual}}$ is covered by a mitigation $swBr$ for ncol. In practice, one desires mitigations that cover many states or mitigate several factors at once. Although it is useful to statically assign severity intervals for analysis purposes, it is obvious that these intervals will in most applications of risk-aware machines have to be estimated online. From a particular state, online estimation allows one to trigger a mitigation based on a severity threshold that codifies an unacceptable level of risk. Moreover, a risk structure makes this estimation explicit and potentially simpler. Overall, the $\mathfrak{R}_{\mathsf{manual}}$ is a simple risk structure, yet showing many features of RISKSTRUCTURES. Figures 11a and 11b in Appendix B show a more complex structure for a set of six factors, generated by YAP in 66 milliseconds. The use of constraints leads to a focus on 24 out of the 4096 (4^6 from using four-phase factors) risk states.

7.5. Discussion: compositional abstraction

Event abstraction Events in the CSP interpretation are atomic observations [Ros10, Ch. 1.5]. The initiation and termination of events representing complex enduring real-world phenomena are to be viewed as non-separable aspects of these events. Consequently, care is necessary when making assumptions about the atomicity of such

[24] GNU/Octave, https://octave.sourceforge.io.

[25] To keep the model simple for the sake of illustration, the upper bound 12 takes into account the possibility that the driver might be able to regain control after the airbag deflates.

events interfering with other events. Nevertheless, we consider the event sets in Fig. 2b and their embedding into compound events (Sect. 7.1) useful to model the *initiation, termination, or other significant events* of the corresponding sub-processes (i.e., endangerment and mitigation processes) of P.

Any implementation of a risk structure as a safety controller needs to make an assumption about how *shared resources*, such as actuators, are used. In CSP, two processes (e.g. two factors) cannot engage in two different events on the same channel at the same time. Hence, resource sharing has to be implemented outside a risk structure, creating two cases: (a) mitigation actions of two factors mapping onto the same resource simultaneously occur and (b) a mitigation action competes with an action outside the safety controller. In case (a), signal *priorities* or more sophisticated signal *merge functions* may determine the control inputs to be forwarded. Case (b) suggests the use of a *safety override*, that is, giving priority to the safety controller.

Shared factor alphabet for compositional abstraction The side conditions in the rules of $\|$ (Definition 15) prohibit any behaviour leading to inconsistent states. The composition constrains the behaviour of two structures with an overlapping scope. Corollary 6 summarises our decisions on defining the constraint operator in such a way that Lemma 16 confirms this operator to form a refinement of a risk structure.

Were we to use arbitrary CSP processes for factors and were we to allow different interfaces for each use of the parallel composition operator, several structures, when composed, would not have guaranteed freedom from interference, exhibit different event and state traces and, consequently, the order of their composition would have led to different risk models. Associative laws for generalised parallel composition in CSP are not universally applicable. Discussions by Oliveira et al. [OCW09] and Roscoe [Ros10, p. 60] highlight how differences in the alphabets shared between each pair in a set of risk structures would entail meaning to the order in which these pairs are composed. For example, in CSP, the equality $(P \underset{X}{\|} Q) \underset{Y}{\|} R = P \underset{X}{\|} (Q \underset{Y}{\|} R)$ holds generally only if $X = Y$. In case of $X \neq Y$, one has to compare the traces of the composed processes to prove specific guarantees to be preserved by their composition. This can be computationally complex.

Overall, we obtain two advantages in RiskStructures when using factors with an alphabet as described in Sect. 7.1. First, all factors have to always agree on their view of the same process P. Second, overlapping scopes in CSP terms then mean copies of factors that will be reduced by idempotency if deterministic (see Lemma 8). The transformation of factors into CSP (Sect. 7.1) encodes all the information from the risk space R into the processes 0^f, f, and $\overline{f}$. This way, we restrict the use of generalised parallel composition to the use of synchronous parallel composition [Ros10, p. 45].

Risk graphs and Yap The risk graphs in Figs. 7, 8 and 10 omit the transition self-loops for nominal, endangered, and mitigated operation because these are non-essential for endangerment and mitigation. Currently, Yap explores risk by changing (i.e., activating, mitigating) one factor at a time unless constraints such as `causes` are used. `causes` enforces the synchronisation of phase changes of involved factors. Because of input-enabledness from the compound event construction (Sect. 7.1), interleaving is avoided, but non-determinism can arise from non-deterministic factors. However, composition according to Definition 15 is more flexible because it allows several factors to change their phases simultaneously. This ability is exploited for the synthesis of safety controllers from Yap models in [GC20]. Furthermore, Definition 15 does not require input-enabledness but it relies on the alphabet defined in Sect. 7.1. Anyway, the use of a risk structure as a concurrent monitor or, more actively, as a safety controller, makes input-enabledness a useful requirement.

7.6. Discussion: uncertainty in RiskStructures

Recall the notion of risk as an action with an *uncertain* and *undesired* outcome. As summarised in Fig. 1, a risk structure focuses on such outcomes.[26] Consequently, we propose risk structures as one way of formalising the irreducible (or aleatory) uncertainty associated with the actions of a risk-aware machine operated in a particular domain. Table 5 highlights *low-level* epistemic and aleatory uncertainties about a risk structure itself as a formal model of aleatory uncertainty in that domain, and as a model of a safety controller of that machine. Table 5 classifies these low-level uncertainties based on the discussion in [BFK13] and indicates how RiskStructures incorporate and support the handling of these uncertainties.

[26] Desired or certain outcomes are not typically subject of operational risk analysis of autonomous machines.

Table 5. Summary of uncertainties inherent to and controllable in RISKSTRUCTURES

RISKSTRUCTURES can be uncertain in their ...	at design-time *epistemic*[1] (imprec.)	at run-time *aleatory*[2] (variab.)	*epistemic*[3] (meas.)	Mathematical approach	Can be implemented by ...
Factor uncertainties					
factor observation	yes	–	yes	subsetting	e^{f} only covers domain subset
impact quantification	yes	yes	yes	intervals	severity bounds
phase changes	yes	–	yes	non-determinism	overlapping factor event sets
risk qualification	yes	–	–	partial order	partial phase order $\preceq_{\mathsf{f}}$
Structural uncertainties					
analysis scope	yes	–	–	relations	factor dependencies/omission
risk qualification	yes	–	–	partial order	partial mitigation order $\precsim_{\mathsf{m}}$
risk quantification	yes	yes	yes	interval arithmetic	strong mitigation order $\preceq_{\mathsf{m}}$
To be investigated in future work					
phase changes	yes	yes	yes	MDPs	transition probabilities

Legend: [1]analytic imprecision, [2]physical parameter variability, [3]parameter measurement, – not applicable

Non-deterministic and partial risk factors With *non-deterministic factors* (Sect. 4.4) in $\mathfrak{R}$, from an observed phase/event, P can reach several distinct phases and, thus, risk states. Without further information, $\mathfrak{R}$ ceases to know which of these states was actually reached. $\mathfrak{R}$ would be in a *risk state region*, ordered and bounded by $\min_{\preceq_{\mathsf{m}}} / \max_{\preceq_{\mathsf{m}}}$ (Sect. 5.4), suggesting to express $\mathfrak{R}$'s uncertainty by a *risk interval*. If factor phases carry information about P's state in terms of disjoint state invariants, we can build state estimators into $\mathfrak{R}$, able to gradually restore lost state information, narrow the interval, and again uniquely identify P's actual risk state. Following up on Sect. 4.4, if we can show that a risk structure is an Input/Output LTS and it can be transformed to a non-deterministic finite Mealy machine, the state estimation problem can be solved, for example, by homing algorithms used for passive testing [NSV03].

Partially specified risk factors (e.g. because of partial activation or mitigation analysis according to Tables 1b and 1c) may lead to *sensible transitions* τ_{c} (cf. Sect. 7.3). Occurrences of τ_{c} could be used to incrementally *learn a property enforcer* guided by unrealised events from risk factors.

8. Related work

Following up Sect. 2.1, we summarise research on dependability modelling for repairable systems, algebraic methods for risk assessment, run-time monitoring, and risk-aware planning and control. We discuss the role of RISKSTRUCTURES in the design of safety controllers.

From dependability analysis to RISKSTRUCTURES Leveson and Stolzy [LS87] investigate reachability graphs of timed Petri nets with failure places to assess *fault-tolerance* of safety-critical real-time systems. RISKSTRUCTURES form a projection and compositional generalisation of such graphs when risk factors are used to model faults. Based on the annotation of a component architecture with a fault model, Unanue et al. [UPM18] show the synthesises of a failure automaton, a temporal fault/repair tree, and MCSs from this tree. From the failure automaton, they construct an extended Petri net to calculate failure probabilities to identify critical MCSs. Unanue et al.'s use of graphical models is intuitive and practical. A failure automaton corresponds to a risk structure with directly reducible risk factors modelling failure and repair. The commonalities of our framework with theirs suggests that RISKSTRUCTURES can use input from state-of-the-art dependability assessments of *repairable systems* for the design of safety controllers. At the moment, however, RISKSTRUCTURES provide no direct support for modelling probabilistic actions.

As discussed in Sect. 6, FTs (Sect. 2.1) can be translated into RISKSTRUCTURES using factor dependencies. Because of the step semantics of constraints (cf. Definition 12), such a translation is also possible for `PAND` and `POR` gates[27] used in dynamic FTs. This way, RISKSTRUCTURES can include a translation of FTs generated from

[27]These are `AND` and `OR` gates that also express priority in a FT by taking into account the order of event occurrences.

system architectures (see, e.g. [UPM18]). The expressiveness of risk factors enables the combination of Boolean factors modelling failures internal to an autonomous robot with factors modelling hazards in physical processes in this robot's environment.

Hansen et al. [HRS98] use the Duration Calculus to derive safety requirements from system FTs for the incremental design of safety-critical real-time controllers. We have shown an integration of FTA and incremental design in [GC17], however, from a less rigorous perspective. Because FTs can be modelled by RiskStructures, an adoption of Hansen et al.'s approach suggests a refinement of the risk factors defined in Figs. 2a and 4 for deriving safety controllers for real-time systems.

Algebraic methods for risk assessment The only algebraic approach to design-oriented risk assessment, we could find, is from Hamdi and Boudriga [HB03]. They formalise IT security risk management as an algebraic datatype specification to check the consistency of security risk analyses viewed as algebras. Probability of occurrence and severity of consequence are modelled as metrics over attack actions to select optimal countermeasures using multi-objective optimisation. While focusing on security management, their algebraic datatype is an inspiration for a further formalisation of consequences and assets in Sect. 6.3. However, our action abstraction and the use of CSP offers a design method for safety controllers as opposed to the more abstract attack model proposed in [HB03], which focuses on risk assessment.

From RiskStructures to safety monitors and controllers As mentioned for the comparison with [UPM18], each factor f (Fig. 2) can be seen as a *sequential monitor automaton* for a process P, with f's events being triggered (i.e., sensors, variable checkers) by P's actions. A risk structure $\mathfrak{R}$ then forms a *concurrent monitor* of P's risk space, each risk state being concurrently monitored for the detection of endangerments (i.e., safety violations) and mitigation successes (i.e., mitigation acceptances).

Guiochet et al. [GPBB08] encode a risk model into a monitor automaton with partially ordered safety modes, each with a constraint limiting the parameters of robot actions to a mode-specific risk level. Refining this idea, Mekki-Mokhtar et al. [MMBG+12] distinguish between safe, warning, and catastrophic states with safety trigger conditions. Based on this framework, Machin et al. [MGW+18] apply model checking to determine whether catastrophic states can be reached despite such triggers. The authors present an algorithm for the synthesis of a mitigation policy, that is, minimal sequences of interventions to reach the nearest safe state from any warning state fulfilling validity, permissiveness, and safety. Our framework could enhance their refined hazard model with an algebraic method and the specification of dependencies between hazards.

Based on the MOP approach to RV [MJG+11], Huang et al. [HEZ+14] present a monitoring infrastructure for (i) checking trace properties (specified in different formalisms) observable from communications between ROS modules and (ii) the property-triggered execution of mitigations. Bogdiukiewicz et al. [BBH+17] describe how monitors can be formally developed in a step-wise manner using Event-B, an abstract state machine language supporting refinement-based development. RiskStructures provide a bridge between formal risk modelling and the design of monitors in Event-B as shown in [BBH+17], in the ROS-specific MOP framework [HEZ+14], or the ROS-based framework proposed in [SLJS16].

From RiskStructures to risk-aware control Works in (stochastic) optimal control [PBHS13, San14, FC14, MS14, SSSS18], summarised in Sect. 2.1, incorporate risk either as a (chance) constraint not to be violated or as a minimisation criterion for determining an optimal plan. The underlying stochastic models allow the per-state estimation of expected risk. Severity intervals would allow a per-state estimation of the *expected risk range*. Complementary to, for example, Sanger's neuronal network-based controller [San14], RiskStructures provide an algebraic method for structuring and composing complex risk models (i.e., state spaces, value functions, action spaces) in a verifiable manner. Once a risk model is found and validated, optimal control provides the low-level mathematical framework for detailed controller design.

9. Conclusion

The certification of autonomous systems requires the rigorous verification of their safety controllers. From the application of formal methods in other domains, we know that the emerging complexity of such controllers can be tackled by compositional verification. With RiskStructures, we proposed a novel algebraic framework for the modelling and analysis of the risk profile of an automated or autonomous machine in its multi-scenario operational environment. This framework also supports the construction of discrete-event safety controllers that,

when integrated into the machine, improve this risk profile. These controllers incorporate an explicit and, thus, verifiable structure of the risk awareness of such a machine. The presented algebraic laws support the design of and reasoning about these controllers. The proposed theory was applied in form of situation/action/factor tables used to craft risk models with the Yap tool and resulted in the generation of risk graphs useful for controller design. To the best of our knowledge, this work is the first to provide an algebraic account of risk modelling and systematic safety controller design for risk-aware machines.

Future work Our work on this framework has led to a range of new research directions we want to pursue in the near future. The further development of our framework will include its extension to more general and more refined forms of risk factors; the elaboration of probabilistic factor semantics; the investigation of more specific mitigation orders, the construction of risk lattices from risk spaces, mitigation orders, and event structures; the development of analysers for risk estimation and synthesizers for safety controllers; the derivation of RiskStructures from dynamical models; the further discussion of factor characteristics such as compatibility; the investigation of distributivity of composition and constraints; and a mechanisation and extension of further proofs using a specialised proof assistant such as Isabelle/UTP [FBC+20].

Acknowledgements

Mario Gleirscher was supported in part by the German Research Foundation (DFG) under the Fellowship Grant no. 381212925. Work by Radu Calinescu and Mario Gleirscher was partially supported by the Lloyd's Register Foundation under the Autonomy Assurance International Programme (AAIP) Grant CSI:Cobot. Radu Calinescu was additionally supported by the UKRI Project EP/V026747/1 "Trustworthy Autonomous Systems Node in Resilience". We would like to thank Simon Foster for inspiring discussions on the use of relational specification; Ana Cavalcanti and Cliff Jones for insightful questions about the abstraction, composition, and methodology underlying RiskStructures; James Baxter, Alvaro Miyazawa, and Pedro Ribeiro for enlightening conversations about CSP. We are also thankful to Sam Clark for helpful feedback on an early version of the introductory and closing sections.

Author Contributions. • **Mario Gleirscher:** Conceptualization, Methodology, Formal analysis and investigation, Writing - original draft preparation, Writing - review and editing, Funding acquisition. • **Radu Calinescu:** Writing - review and editing, Funding acquisition, Resources, Supervision. • **Jim Woodcock:** Formal analysis and investigation, Writing - review and editing, Resources, Supervision.

Abbreviations

AV	Autonomous vehicle
CSL	Continuous stochastic logic
CSP	Communicating sequential processes
CTMC	Continuous time Markov chain
ETA	Event tree analysis
FMEA	Failure mode and effects analysis
FT	Fault tree

FTA	Fault tree analysis
HazOp	Hazard and operability studies
IH	Induction hypothesis
IOLTS	Input/Output LTS
IS	Induction step
LOPA	Layer of protection analysis
LTS	Labelled transition system
MCS	Minimum cut set
MDP	Markov decision process
MOP	Monitoring-oriented programming
ROS	Robot operating system
RV	Run-time verification
STAMP	System-theoretic accident model and process
STPA	System-theoretic process analysis
TS	Transition system
UML	Unified modelling language

A. Proof details

This section provides further proof details. The symbol ◪ indicates a discharged case distinction.

Proof of Lemma 1. The proof is by mutual existence and uniqueness: For each $\sigma \in R(F_1 \cup F_2)$ (i) there exists a $\sigma_1 \cup \sigma_2 \in R(F_1) \otimes R(F_2)$ and (ii) this pair is unique, and (iii, iv) conversely.

Below, we will need the infix binary operator scheme $\cdot\,|_{F'} \colon R(F) \to R(F')$ that describes the (phase-invariant) projection from $R(F)$ to $R(F')$ with $F' \subseteq F$.

We show (i): Let $\sigma \in R(F_1 \cup F_2)$, then, by definition, σ is a total injection and, hence, every restriction of σ is a total injection, particularly, the restrictions $\sigma\,|_{F_1}$ and $\sigma\,|_{F_2}$. Obviously, we have $\sigma\,|_{F_1}(\mathsf{f}) = \sigma\,|_{F_2}(\mathsf{f})$ for all $\mathsf{f} \in F_1 \cap F_2$. Furthermore, by definition, $\sigma(\mathsf{f})$ is faithful to Ph_{f} for all $\mathsf{f} \in F_1 \cup F_2$ and, thus, so are both these restrictions. These two results lead to $\sigma_1 = \sigma\,|_{F_1} \in R(F_1)$ and $\sigma_2 = \sigma\,|_{F_2} \in R(F_2)$ and, finally, to the existence of the wanted pair. ◪

We show (ii): Suppose there are two pairs $(\sigma_1, \sigma_2) \neq (\sigma_1', \sigma_2) \in R(F_1) \otimes R(F_2)$. Then there exists $\mathsf{f} \in F_1 \setminus F_2$ where $\sigma_1(\mathsf{f}) \neq \sigma_1'(\mathsf{f})$, thus, also $\sigma_1 \cup \sigma_2 \neq \sigma_1' \cup \sigma_2$. However, there can be no faithful total injection $\sigma \in R(F_1 \cup F_2)$ such that $\sigma = \sigma_1 \cup \sigma_2 \wedge \sigma = \sigma_1' \cup \sigma_2$. ◪

We show (iii): For each $\sigma_1 \cup \sigma_2 \in R(F_1) \otimes R(F_2)$ there exists a $\sigma \in R(F_1 \cup F_2) = \{\sigma \in (F_1 \cup F_2) \to \bigcup_{\mathsf{f} \in F_1 \cup F_2} Ph_{\mathsf{f}} \mid \sigma \text{ is a total injection} \wedge \forall \mathsf{f} \in F_1 \cup F_2 \colon \sigma(\mathsf{f}) \in Ph_{\mathsf{f}}\}$: By definition, both σ_1 and σ_2 are faithful total injections (i.e., matching results for all $\mathsf{f} \in F_1 \cap F_2$). Then, we can construct a faithful total injection σ by applying set union to the domains and co-domains of these two. Thus, $\sigma \in R(F_1 \cup F_2)$. ◪

We show (iv): Suppose from $\sigma_1 \cup \sigma_2$ we can construct $\sigma \neq \sigma' \in R(F_1 \cup F_2)$ then there exists $\mathsf{f} \in F_1 \cup F_2$ such that $\sigma(\mathsf{f}) \neq \sigma'(\mathsf{f})$. However, then either σ_1 or σ_2 must have violated injectivity which in turn would have violated the definition of $R(F_1) \otimes R(F_2)$. □

Proof of Lemma 2. $(\mathcal{F}, \cup)$ is a semi-group because $\cup$ is an associative binary operation on $\mathcal{F}$. We have that

$$\sigma_1 \approx (\sigma_2 \cup \sigma_3) \Leftrightarrow \sigma_1 \approx \sigma_2 \wedge \sigma_1 \approx \sigma_3\ . \tag{17}$$

(The sub-proof based on the definition of $\approx$ is omitted here.) Based on Formula (17), we show by algebraic manipulation that the binary operation $\otimes$ on $\mathcal{R}$ is associative:

$$
\begin{aligned}
&R(F_1) \otimes (R(F_2) \otimes R(F_3)) \\
&= \{\sigma_1 \cup \sigma \mid \sigma_1 \in R(F_1) \wedge \sigma \in R(F_2) \otimes R(F_3) \wedge \sigma_1 \approx \sigma\} \\
&= \{\sigma_1 \cup (\sigma_2 \cup \sigma_3) \mid \sigma_1 \in R(F_1) \wedge (\sigma_2 \cup \sigma_3) \in R(F_2) \otimes R(F_3) \wedge \sigma_1 \approx (\sigma_2 \cup \sigma_3)\} \\
&= \{\sigma_1 \cup (\sigma_2 \cup \sigma_3) \mid \sigma_1 \in R(F_1) \wedge (\sigma_2 \in R(F_2) \wedge \sigma_3 \in R(F_3) \wedge \sigma_2 \approx \sigma_3) \wedge \sigma_1 \approx (\sigma_2 \cup \sigma_3)\} \\
&= \{(\sigma_1 \cup \sigma_2) \cup \sigma_3 \mid (\sigma_1 \in R(F_1) \wedge \sigma_2 \in R(F_2)) \wedge \sigma_3 \in R(F_3) \wedge \sigma_2 \approx \sigma_3 \wedge \sigma_1 \approx (\sigma_2 \cup \sigma_3)\} \quad \text{(by Formula (17))} \\
&= \{(\sigma_1 \cup \sigma_2) \cup \sigma_3 \mid (\sigma_1 \in R(F_1) \wedge \sigma_2 \in R(F_2)) \wedge \sigma_3 \in R(F_3) \wedge \sigma_2 \approx \sigma_3 \wedge \sigma_1 \approx \sigma_2 \wedge \sigma_1 \approx \sigma_3\} \\
&= \{(\sigma_1 \cup \sigma_2) \cup \sigma_3 \mid (\sigma_1 \in R(F_1) \wedge \sigma_2 \in R(F_2) \wedge \sigma_1 \approx \sigma_2) \wedge \sigma_3 \in R(F_3) \wedge (\sigma_1 \cup \sigma_2) \approx \sigma_3\} \\
&= \{(\sigma_1 \cup \sigma_2) \cup \sigma_3 \mid (\sigma_1 \cup \sigma_2) \in R(F_1) \otimes R(F_2) \wedge \sigma_3 \in R(F_3) \wedge (\sigma_1 \cup \sigma_2) \approx \sigma_3\} \\
&= (R(F_1) \otimes R(F_2)) \otimes R(F_3)
\end{aligned}
$$

Hence, $(\mathcal{R}, \otimes)$ is a semi-group, too. Lemma 1 completes the proof. □

Proof of Lemma 3.

We express the basic factor template from Fig. 2a in CSP as

$$
\begin{aligned}
\mathsf{f}_{(2a)} &= 0^f \\
0^f &= ?x : e^{\mathsf{f}} \to f \,\Box\, ?x : o_n^{\mathsf{f}} \to 0^f \\
f &= ?x : m_d^{\mathsf{f}} \to 0^f \,\Box\, ?x : o_e^{\mathsf{f}} \to f \,\Box\, ?x : m^{\mathsf{f}} \to \overline{f} \\
\overline{f} &= ?x : \overline{e}^{\mathsf{f}} \to f \,\Box\, ?x : o_m^{\mathsf{f}} \to \overline{f} \,\Box\, ?x : m_r^{\mathsf{f}} \to 0^f \ .
\end{aligned}
$$

(As indicated in the Sect. 4.1 and 4.4, we consistently omit $\underline{e}^{\mathsf{f}}$ for the sake of simplicity.) Analogously, we express the factor template with isolated non-determinism from Fig. 4 in CSP as

$$
\begin{aligned}
\mathsf{f}_{(4)} &= 0^f \\
0^f &= ?x : e^{\mathsf{f}} \setminus ue^{\mathsf{f}} \to f \,\Box\, ?x : o_n^{\mathsf{f}} \setminus ue^{\mathsf{f}} \to 0^f \,\Box\, \big(?x : ue^{\mathsf{f}} \to (0^f \sqcap f)\big) \\
f &= ?x : m_d^{\mathsf{f}} \to 0^f \,\Box\, ?x : o_e^{\mathsf{f}} \setminus um^{\mathsf{f}} \to f \,\Box\, ?x : m^{\mathsf{f}} \setminus um^{\mathsf{f}} \to \overline{f} \,\Box\, \big(?x : um^{\mathsf{f}} \to (f \sqcap \overline{f})\big) \\
\overline{f} &= ?x : \overline{e}^{\mathsf{f}} \to f \,\Box\, ?x : o_m^{\mathsf{f}} \setminus um_r^{\mathsf{f}} \to \overline{f} \,\Box\, ?x : m_r^{\mathsf{f}} \setminus um_r^{\mathsf{f}} \to 0^f \,\Box\, \big(?x : um_r^{\mathsf{f}} \to (0^f \sqcap \overline{f})\big).
\end{aligned}
$$

The inherent difference of the actions m_d^{f}, $\overline{e}^{\mathsf{f}}$, and $\underline{e}^{\mathsf{f}}$ from their phase competitors allows us to make the simplifying assumption $\overline{e}^{\mathsf{f}} \cap o_m^{\mathsf{f}} = \overline{e}^{\mathsf{f}} \cap m_r^{\mathsf{f}} = \underline{e}^{\mathsf{f}} \cap o_e^{\mathsf{f}} = \underline{e}^{\mathsf{f}} \cap m^{\mathsf{f}} = \underline{e}^{\mathsf{f}} \cap m_d^{\mathsf{f}} = m^{\mathsf{f}} \cap m_d^{\mathsf{f}} = \varnothing$. We further define $ue^{\mathsf{f}} = o_n^{\mathsf{f}} \cap e^{\mathsf{f}}$, $um^{\mathsf{f}} = o_e^{\mathsf{f}} \cap m^{\mathsf{f}}$, $um_r^{\mathsf{f}} = o_m^{\mathsf{f}} \cap m_r^{\mathsf{f}}$. For the equivalence proof, we have to distinguish whether or not the events in the three sets $\{e^{\mathsf{f}}, o_n^{\mathsf{f}}\}$, $\{m_d^{\mathsf{f}}, o_e^{\mathsf{f}}, m^{\mathsf{f}}\}$, and $\{\overline{e}^{\mathsf{f}}, o_m^{\mathsf{f}}, m_r^{\mathsf{f}}\}$ are pairwise disjoint.

Case 1: Events are pairwise disjoint. Then,

1. from the emptiness of ue^{f}, um^{f}, and um_r^{f},
2. from applying the CSP laws $?x : \varnothing \to P =_{FD} STOP$ and $P \,\Box\, STOP =_{FD} P$ [Sch99, p. 93] to $\mathsf{f}_{(4)}$, and
3. from the resulting structure of $\mathsf{f}_{(4)}$ being equivalent to $\mathsf{f}_{(2a)}$,

we immediately derive $\mathsf{f}_{(4)} =_{FD} \mathsf{f}_{(2a)}$. ◪

Case 2: Events are not pairwise disjoint. Then,

1. from the potential non-emptiness of ue^{f}, um^{f}, and um_r^{f},
2. from applying the step law of external choice [Sch99, p. 93] to $\mathsf{f}_{(2a)}$ in order to isolate, for example, $?x : o_n^{\mathsf{f}} \cap e^{\mathsf{f}} \to (0^f \sqcap f)$, and
3. from our definitions (e.g. $ue^{\mathsf{f}} = o_n^{\mathsf{f}} \cap e^{\mathsf{f}}$),

we again immediately have that $\mathsf{f}_{(2a)} =_{FD} \mathsf{f}_{(4)}$. The cases 1 and 2 complete the proof. □

The following corollary and its proof rule out potential errors in the proof of Lemma 4.

Corollary 7 (Converse of Lemma 4)

$$\neg(\sigma \preceq_{\mathsf{m}} \sigma') \Leftarrow \neg(\sigma \precsim_{\mathsf{m}} \sigma') \qquad \text{(by definition of } \not\preceq_{\mathsf{m}}, \not\precsim_{\mathsf{m}})$$

$$\sigma \not\preceq_{\mathsf{m}} \sigma' \Leftarrow \sigma \not\precsim_{\mathsf{m}} \sigma' \qquad \text{(by negation and definition of } \preceq_{\mathsf{m}}, \precsim_{\mathsf{m}})$$

$$\sigma \succ_{\mathsf{m}} \sigma' \vee \big((\sigma,\sigma') \notin \preceq_{\mathsf{m}} \wedge\ (\sigma',\sigma) \notin \preceq_{\mathsf{m}}\big) \Leftarrow \sigma \succ^{\sim}_{\mathsf{m}} \sigma' \vee \big((\sigma,\sigma') \notin \precsim_{\mathsf{m}} \wedge\ (\sigma',\sigma) \notin \precsim_{\mathsf{m}}\big) \tag{18}$$

Proof of Corollary 7. Case 1: If $(\sigma,\sigma') \notin \precsim_{\mathsf{m}} \wedge\ (\sigma',\sigma) \notin \precsim_{\mathsf{m}}$ then (by Definitions 8 and 9) also $(\sigma,\sigma') \notin \preceq_{\mathsf{m}} \wedge (\sigma',\sigma) \notin \preceq_{\mathsf{m}}$ and, therefore, Formula (18). ◪

Case 2: If $\sigma \succ^{\sim}_{\mathsf{m}} \sigma' = \sigma' \precsim_{\mathsf{m}} \sigma \wedge \sigma' \neq^{\sim}_{\mathsf{m}} \sigma$, then we have either $(\sigma,\sigma') \notin \preceq_{\mathsf{m}} \wedge\ (\sigma',\sigma) \notin \preceq_{\mathsf{m}}$ (because of some incomparable phases) which fulfils Formula (18). Alternatively, we have $(\sigma',\sigma) \in \preceq_{\mathsf{m}}$ which means σ and σ' are fully comparable and, because of Corollary 2, we have $\sigma \succ_{\mathsf{m}} \sigma'$. □

Proof of Lemma 5. For this we only need to show that any two risk states $\sigma, \sigma' \in R$ are (i) comparable and (ii) antisymmetric: $[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}} \wedge [\sigma']_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma]_{\sim_{\mathsf{s}}} \Rightarrow [\sigma]_{\sim_{\mathsf{s}}} =_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$.

We show (i) by showing that the conditions (a) and (b) guarantee the comparability of any two risk states in R based on the interval order $\leq$ as defined above and use the fact that comparability can be dropped from $R/\sim_{\mathsf{s}}$ to R using $[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}} \iff \forall \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}} : \check{\sigma} \leq_{\mathsf{m}} \check{\sigma}'$ (Formula (6)). Recall that we equate the empty interval with the empty set, $[) = \varnothing$. Now, we have to consider the following three cases to complete the sub-proof of (i):

Case "no risk factors activated": $[) \geq [) \vee \varnothing \subset \varnothing$ validates (b) and lack of comparability invalidates (a), yet we have $[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$. ◪

Case "at least one risk factor activated only in $\check{\sigma}$": Let $[a,b) = S(\check{\sigma})$. $[a,b) \geq [) \vee [a,b) \subset \varnothing$ invalidates (a) and (b). However, the observation $[) \geq [a,b) \vee \varnothing \subset [a,b)$ yields $[\sigma']_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma]_{\sim_{\mathsf{s}}}$. The dual of this case works analogously. ◪

Case "at least one risk factor activated in both $\check{\sigma}, \check{\sigma}'$": Let $[a,b) = S(\check{\sigma})$ and $[c,d) = S(\check{\sigma}')$. We need to show $[a,b) \geq [c,d) \vee [a,b) \subset [c,d)$ for all $a, b, c, d \in \mathbb{R}$: We can assume $a \leq b \wedge c \leq d$ by definition of intervals. Then, we face the case that

- $c \leq a \wedge d \leq b$ validates (a) by definition of $\leq$ over intervals,
- $c > a \wedge d \leq b$ validates (b),
- $c \leq a \wedge d > b$ implies (b) for the dual case $[\sigma']_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma]_{\sim_{\mathsf{s}}}$, or
- $c > a \wedge d > b$ implies (a) for the dual case.

Hence, from each of these four cases, either $[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$ or $[\sigma']_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma]_{\sim_{\mathsf{s}}}$ follows.

Then, we show (ii) by contradiction: Assuming $[\sigma]_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}} \wedge [\sigma']_{\sim_{\mathsf{s}}} \leq_{\mathsf{m}} [\sigma]_{\sim_{\mathsf{s}}}$, we have $\forall \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}} : \check{\sigma} \leq_{\mathsf{m}} \check{\sigma}' \wedge \check{\sigma} \geq_{\mathsf{m}} \check{\sigma}'$ by dropping from $R/\sim_{\mathsf{s}}$. Now, we claim that $[\sigma]_{\sim_{\mathsf{s}}} \neq_{\mathsf{m}} [\sigma']_{\sim_{\mathsf{s}}}$. Hence, $\exists \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}} : \check{\sigma} \neq_{\mathsf{m}} \check{\sigma}'$ and, consequently, $\exists \check{\sigma} \in [\sigma]_{\sim_{\mathsf{s}}}, \check{\sigma}' \in [\sigma']_{\sim_{\mathsf{s}}} : \check{\sigma} \not\leq_{\mathsf{m}} \check{\sigma}' \vee \check{\sigma} \not\geq_{\mathsf{m}} \check{\sigma}'$. The latter contradicts our assumption. □

Proof of Lemma 7. Fix a finite $F \subseteq \mathcal{F}$ and a pair $\sigma, \sigma' \in R(F)$. The proof is by induction over F and relies on the assumption that, for any $\mathsf{f} \in F$, the phase f is the *unique maximal element* of $\preceq_{\mathsf{f}}$ and that $active$ only returns such elements:

Induction start $F_0 = \varnothing$: $\sigma\,|_{\varnothing} \preceq_{\mathsf{m}} \sigma'\,|_{\varnothing}$ holds trivially and so does $active(\sigma'\,|_{\varnothing}) \subseteq active(\sigma\,|_{\varnothing})$.

Induction step (IS) $F_{n+1} = F_n \cup \{\mathsf{f}\}$ where $n \geq 0$ and $\mathsf{f} \in F \setminus F_n$: For the IH, assume $\sigma\,|_{F_n} \preceq_{\mathsf{m}} \sigma'\,|_{F_n} \Rightarrow active(\sigma'\,|_{F_n}) \subseteq active(\sigma\,|_{F_n})$. Then, we show that $\sigma\,|_{F_{n+1}} \preceq_{\mathsf{m}} \sigma'\,|_{F_{n+1}} \Rightarrow active(\sigma'\,|_{F_{n+1}}) \subseteq active(\sigma\,|_{F_{n+1}})$ (i.e., the IS). For this, we prove

- the case of *incomparable phases* $(\sigma(\mathsf{f}), \sigma'(\mathsf{f})) \notin \preceq_{\mathsf{f}}$: In this case, the state pair gets incomparable and the implication is trivially fulfilled,
- the *inverse case* $\sigma(\mathsf{f}) \succ_{\mathsf{f}} \sigma'(\mathsf{f})$: In this case, the state pair gets incomparable and the implication is again trivially fulfilled, and
- the *aligned case* $\sigma(\mathsf{f}) \preceq_{\mathsf{f}} \sigma'(\mathsf{f})$: In this case, the state pair stays comparable. However, $\sigma'(\mathsf{f})$ can either be f (hence, $\sigma(\mathsf{f}) = f$ and maintaining $\subseteq$), $\overline{f}$ (hence, $\sigma(\mathsf{f}) \in \{f, \overline{f}, 0^f\}$ and maintaining $\subseteq$), or 0^f (see former sub-case).

Having proved these cases completes the induction step by establishing the IH.

For $\precsim_{\mathsf{m}}$, we only have to substitute case 1 of the IS: For f, the smallest $\preceq_{\mathsf{f}}$ after Definition 3 implies incomparability at most for $(0^f, \overline{f})$ and $(\overline{f}, 0^f)$. These two phase pairs of f would not alter the $active$ maps of both states and hence maintain $\subseteq$ as well. □

Proof of Lemma 12. From associativity of set intersection in Definition 12, we know that the order in which constraints are applied to $R \times R$ does not matter. Consequently, we also have $[\![k]\!]^R_{\mathsf{c}} \supseteq [\![\{k, k'\}]\!]^R_{\mathsf{c}}$ for any $k, k' \in \mathcal{C}$, that is, constraints only prune $R \times R$. First, we prove the lemma

$$[[\mathfrak{R}]_{\{k\}}]_C = [\mathfrak{R}]_{\{k\}\cup C} \,. \tag{19}$$

Fix $\sigma \xrightarrow{e} \sigma'$.

$\Rightarrow$:

1. *Case* $(\sigma, \sigma') \in [\![k]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-step applies to $[\mathfrak{R}]_{\{k\}}$.
   (a) *Case* $(\sigma, \sigma') \in [\![C]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-step applies also to $[[\mathfrak{R}]_{\{k\}}]_C$. Then, (σ, σ') must be in $[\![k]\!]^R_{\mathsf{c}} \cap [\![C]\!]^R_{\mathsf{c}}$, which by Definition 12 is $[\![\{k\} \cup C]\!]^R_{\mathsf{c}}$. Because of $(\sigma, \sigma') \in [\![\{k\} \cup C]\!]^R_{\mathsf{c}}$, $[\cdot]_C$-step applies to $[\mathfrak{R}]_{\{k\}\cup C}$ in the same way as it does to $[[\mathfrak{R}]_{\{k\}}]_C$. $[\cdot]_C$-cancel does not apply. Consequently, if $\sigma \xrightarrow{e} \sigma'$ is permitted (and added) by the left-hand side (LHS) then it is permitted by the right-hand side (RHS) of Formula (19).
   (b) *Case* $(\sigma, \sigma') \notin [\![C]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-cancel applies to $[[\mathfrak{R}]_{\{k\}}]_C$ instead of $[\cdot]_C$-step, resulting in the production of $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ rather than $\sigma \xrightarrow{e}_C \sigma'$ on the LHS. Because $(\sigma, \sigma') \notin [\![C]\!]^R_{\mathsf{c}}$, we have $(\sigma, \sigma') \notin [\![\{k\}]\!]^R_{\mathsf{c}} \cap [\![C]\!]^R_{\mathsf{c}}$ and, by Definition 12, $(\sigma, \sigma') \notin [\![\{k\} \cup C]\!]^R_{\mathsf{c}}$. Hence, $[\cdot]_C$-cancel applies to RHS and produces $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$.
2. *Case* $(\sigma, \sigma') \notin [\![k]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-cancel applies to $[\mathfrak{R}]_{\{k\}}$ instead of $[\cdot]_C$-step, producing $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ instead of $\sigma \xrightarrow{e}_C \sigma'$ on the inner LHS.
   (a) *Case* $(\sigma, \sigma) \in [\![C]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-step is applicable to $[[\mathfrak{R}]_{\{k\}}]_C$, producing $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ for the outer LHS.
   (b) *Case* $(\sigma, \sigma) \notin [\![C]\!]^R_{\mathsf{c}}$: $[\cdot]_C$-cancel is applicable to $[[\mathfrak{R}]_{\{k\}}]_C$, producing $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ for the outer LHS.

   Because of $[\![\{k\} \cup C]\!]^R_{\mathsf{c}} = [\![k]\!]^R_{\mathsf{c}} \cap [\![C]\!]^R_{\mathsf{c}}$, in both cases (a) and (b), the $[\cdot]_C$-cancel rule applies to $[\mathfrak{R}]_{\{k\}\cup C}$, producing a $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ on the RHS of Formula (19). ◪

$\Leftarrow$:

1. If $(\sigma, \sigma') \in [\![\{k\} \cup C]\!]^R_{\mathsf{c}}$ then $[\cdot]_C$-step is applicable to $[\mathfrak{R}]_{\{k\}\cup C}$ and, because of $[\![k]\!]^R_{\mathsf{c}} \cap [\![C]\!]^R_{\mathsf{c}}$, to the inner and outer constraint in $[[\mathfrak{R}]_{\{k\}}]_C$. Thus, if $\sigma \xrightarrow{e}_C \sigma'$ is produced on the RHS then so on the LHS.
2. If $(\sigma, \sigma') \notin [\![\{k\} \cup C]\!]^R_{\mathsf{c}}$ then $[\cdot]_C$-cancel applies to $[\mathfrak{R}]_{\{k\}\cup C}$ and produces $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ on the RHS. However, because of $(\sigma, \sigma') \notin [\![k]\!]^R_{\mathsf{c}} \cap [\![C]\!]^R_{\mathsf{c}}$, we have at least one of the two cases for the LHS:
   (a) *Case* $(\sigma, \sigma') \notin [\![k]\!]^R_{\mathsf{c}}$: This matches case (2a), that is, $[\cdot]_C$-cancel and $[\cdot]_C$-step produce $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ for $[[\mathfrak{R}]_{\{k\}}]_C$.
   (b) *Case* $(\sigma, \sigma') \notin [\![C]\!]^R_{\mathsf{c}}$: This matches either case (1b) or case (2b), that is, in both cases $[\cdot]_C$-cancel and $[\cdot]_C$-step produce $\sigma \xrightarrow{\tau_{\mathsf{f}}}_C \sigma$ for $[[\mathfrak{R}]_{\{k\}}]_C$. ◪

Second, we show by induction over C_1 that the $[\cdot]_C$-step and $[\cdot]_C$-cancel rules apply in the same way to both sides of Formula (15).

Induction starts with $C_1^0 = \varnothing$: In this case, we have $[[\mathfrak{R}]_{\varnothing}]_{C_2} = [\mathfrak{R}]_{\varnothing\cup C_2}$ and by applying $[\cdot]_C$-step or Definition 12 on the left and $\varnothing \cup C_2 = C_2$ on the right, we get the tautology $[\mathfrak{R}]_{C_2} = [\mathfrak{R}]_{C_2}$. ◪

IS with $C_1^{n+1} = C_1^n \cup \{k\}$ where $n \geq 0$ and $k \in C_1 \setminus C_1^n$: By assuming $[[\mathfrak{R}]_{C_1^n}]_{C_2} = [\mathfrak{R}]_{C_1^n \cup C_2}$ (IH), we show

$$\begin{aligned}
&[[\mathfrak{R}]_{C_1^{n+1}}]_{C_2} && \text{(by def of IS)}\\
&= [[\mathfrak{R}]_{C_1^n \cup \{k\}}]_{C_2} && \text{(by $\cup$-comm and Formula (19))}\\
&= [[[\mathfrak{R}]_{\{k\}}]_{C_1^n}]_{C_2} && \text{(by IH)}\\
&= [[\mathfrak{R}]_{\{k\}}]_{C_1^n \cup C_2} && \text{(by Formula (19))}\\
&= [\mathfrak{R}]_{\{k\}\cup C_1^n \cup C_2} && \text{(by def)}\\
&= [\mathfrak{R}]_{C_1^{n+1} \cup C_2} && \square
\end{aligned}$$

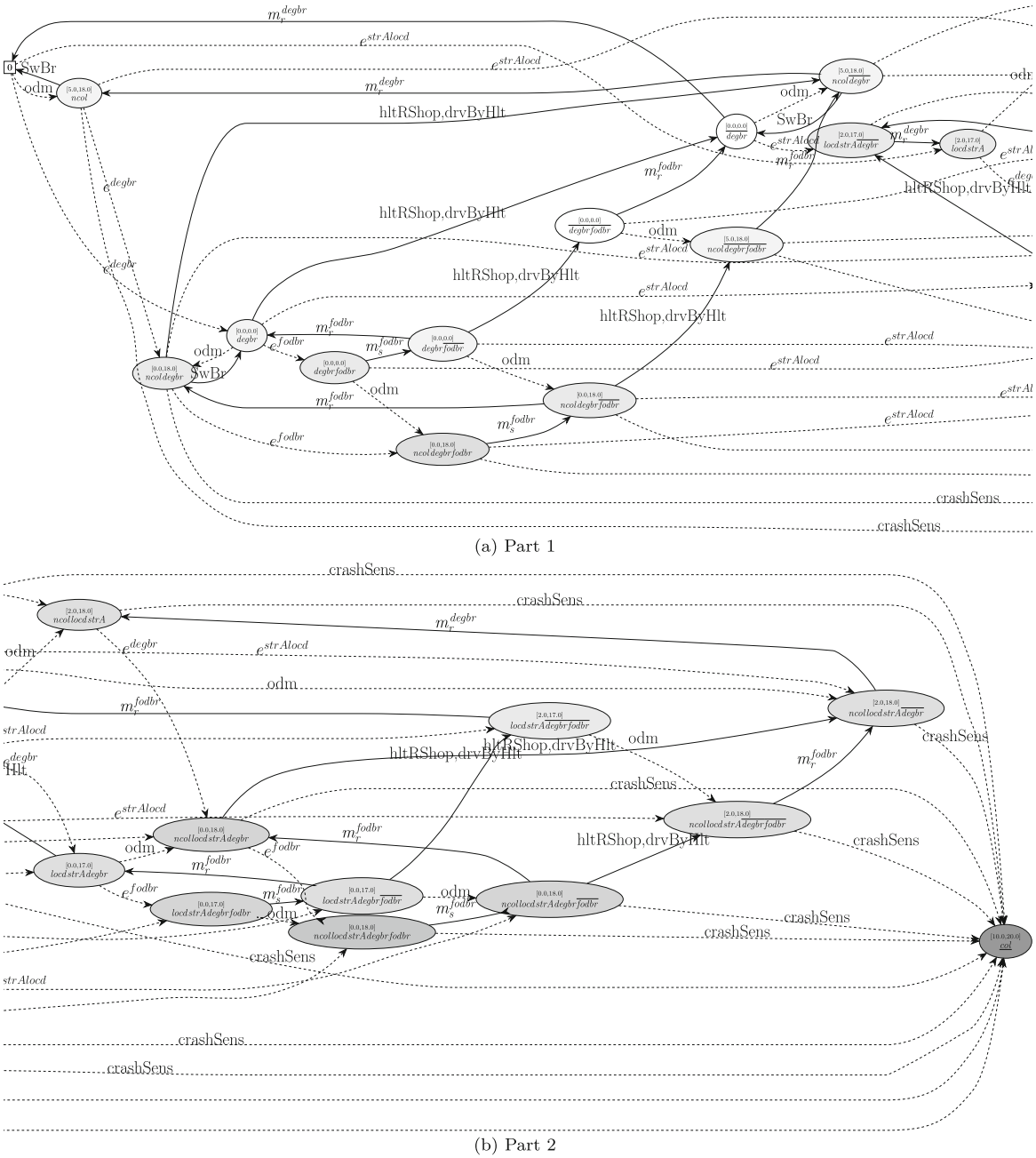


Fig. 11. Risk structure for all factors mentioned in Sects. 4.3, 7.4

B. Supplemental material for the road vehicle example

Figures 11a and 11b contain two parts of a larger risk structure for the road vehicle example illustrated and discussed in the Sects. 7.4, 4.3, 3.

References

[AASB+06] Alami R, Albu-Schäffer A, Bicchi A, Bischoff R, Chatila R, De Luca A, De Santis A, Giralt G, Guiochet J, Hirzinger G, Ingrand F, Lippiello V, Mattone R, Powell D, Sen S, Siciliano B, Tonietti G, Villani L (2006) Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In: Intelligent robots and systems (IROS), IEEE/RSJ international conference

[AKWB11] Althoff D, Kuffner JJ, Wollherr D, Buss M (2011) Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. Auton Robots 32(3):285–302

[ALRL04] Avizienis A, Laprie J-C, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. Dependable Secure Comput IEEE Trans 1(1):11–33

[AZI+17] Ajoudani A, Zanchettin AM, Ivaldi S, Albu-Schäffer A, Kosuge K, Khatib O (2017) Progress and prospects of the human-robot collaboration. Auton Robots, 42(5):957–975

[BBH+17] Bogdiukiewicz C, Butler M, Hoang TS, Paxton M, Snook J, Waldron X, Wilkinson T (2017) Formal development of policing functions for intelligent systems. In: Software Reliability Engineering (ISSRE), 28th IEEE international symposium

[BCS07] Boudali H, Crouzen P, Stoelinga M (2007) Dynamic fault tree analysis using input/output interactive Markov chains. In: Dependable systems and networks (DSN), 37th annual IEEE/IFIP international conference, pp 708–717

[BFK13] Beer M, Ferson S, Kreinovich V (2013) Imprecise probabilities in engineering analyses. Mech Syst Signal Process, 37(1-2):4–29

[Bir17] Birolini A (2017) Reliability Engineering. Springer, 8th edition

[BK08] Baier C, Katoen J-P (2008) Principles of Model Checking. MIT Press

[BS01] Broy M, Stølen K (2001) Specification and Development of Interactive Systems. Springer

[CLC+19] Chen C, Liu X, Chen H-H, Li M, Zhao L (2019) A rear-end collision risk evaluation and control scheme using a Bayesian network model. IEEE Trans Intell Transp Syst, 20(1):264–284

[Eri15] Ericson CA (2015) Hazard analysis techniques for system safety. Wiley, 2nd edition

[FA04] Fraichard T, Asama H (2004) Inevitable collision states – a step towards safer robots?. Adv Robotics, 18(10):1001–1024

[FBC+20] Foster S, Baxter J, Cavalcanti A, Woodcock J, Zeyda F (2020) Unifying semantic foundations for automated verification tools in Isabelle/UTP. Sci Comput Program, 197:102510

[FC14] Feyzabadi S, Carpin S (2014) Risk-aware path planning using hierarchical constrained Markov decision processes. In: Automation science and engineering (CASE), IEEE international conference IEEE

[FGC20] Foster S, Gleirscher M, Calinescu R (2020) Towards deductive verification of control algorithms for autonomous marine vehicles. In: Engineering of complex computer systems (ICECCS), 25th international conference, pp 113–118, Singapore

[Foo78] Foot P (1978) The problem of abortion and the doctrine of the double effect. Virtues and Vices and Other Essays in Moral Philosopy, 19. Originally published in 1967.

[GC17] Gleirscher M, Carlan C (2017) Arguing from hazard analysis in safety cases: A modular argument pattern. In: High Assurance Systems Engineering (HASE), 18th international symposium, pp 53–60. IEEE

[GC20] Gleirscher M, Calinescu R (2020) Safety controller synthesis for collaborative robots. In: Engineering of complex computer systems (ICECCS), 25th international conference, pp 83–92, Singapore

[GFW19] Gleirscher M, Foster S, Woodcock J (2019) New opportunities for integrated formal methods. ACM Comput Surv, 52:117:1–117:36

[GK17] Gleirscher M, Kugele S (2017) From hazard analysis to hazard mitigation planning: The automated driving case. In: Barrett C, et al., editor, NASA Formal Methods (NFM), 9th international symposium, volume 10227 of *LNCS*. Springer

[Gle14] Gleirscher M (2014) Behavioral Safety of Technical Systems. Dissertation, Technische Universität München

[Gle17] Gleirscher M (2017) Run-time risk mitigation in automated vehicles: A model for studying preparatory steps. In: Bulwahn L, Kamali M, Linker S, (eds), Formal Verification of Autonomous Vehicles (FVAV), 1st iFM Workshop, EPTCS

[Gle18] Gleirscher M (2018) Strukturen für die Gefahrenerkennung und -behandlung in autonomen Maschinen. In: Jürgen B, Petra W (eds), Beiträge zu einer Systemtheorie Sicherheit, acatech DISKUSSION, Chapter 8.4, pp 154–167. Herbert Utz Verlag, München

[Gle20] Gleirscher M (2020) Yap: Tool support for deriving safety controllers from hazard analysis and risk assessments. In: Luckuck M, Farrell M (eds), Formal Methods for Autonomous Systems (FMAS), 2nd Workshop, volume 329 of EPTCS, pp 31–47. Open Publishing Association

[Gle21] Gleirscher M (2021) Yap Against Perils: Application Guide and User's Manual. University of York and Technical University of Munich

[GM20] Gleirscher M, Marmsoler D (2020) Formal methods in dependable systems engineering: A survey of professionals from Europe and North America. Empir Softw Eng, 25(6):4473–4546

[GMGP10] Guiochet J, Martin-Guillerez D, Powell D (2010) Experience with model-based user-centered risk assessment for service robots. In: High Assurance Systems Engineering (HASE), 12th IEEE international symposium

[GPBB08] Guiochet J, Powell D, Baudin É, Blanquart J-P(2008) Online safety monitoring using safety modes. In: Technical challenges for dependable robots in human environments, workshop, pp 1–13. Rapport LAAS No. 08339

[Har00] Harpwood V (2000) Principles of Tort Law. Cavendish, 4th edn

[HASH09] Haddadin S, Albu-Schäffer A, Hirzinger G (2009) Requirements for safe robots: Measurements, analysis and new insights. Int J Robot Res, 28(11-12):1507–1527

[HB03] Hamdi M, Boudriga N (2003) Algebraic specification of network security risk management. In: ACM workshop on formal methods in security engineering (FMSE). ACM Press

[HEZ+14] Huang J, Erdogan C, Zhang Y, Moore B, Luo Q, Sundaresan A, Rosu G (2014) ROSRV: Runtime verification for robots. In: Runtime verification, pp 247–254. Springer

[HG03] Holland O, Goodman R (2003) Robots with internal models: A route to machine consciousness?. J Conscious Stud, 10(4-5):77–109

[HM99] Howe RD, Matsuoka Y (1999) Robotics for surgery. Ann Rev Biomed Eng, 1(1):211–240

[Hoa85] Hoare T (1985) Communicating sequential processes. International series in computer science, Prentice-Hall
[HRS98] Hansen KM, Ravn AP, Stavridou V (1998) From safety analysis to software requirement. IEEE Trans Softw Eng, 24(7):573–84
[IMM18] Iamsumang C, Mosleh A, Modarres M (2018) Monitoring and learning algorithms for dynamic hybrid Bayesian network in on-line system health management applications. Reliab Eng Syst Safety, 178:118–129
[IT90] Ishibuchi H, Tanaka H (1990) Multiobjective programming in optimization of the interval objective function. Eur J Oper Res, 48(2):219–225
[KG81] Kaplan S, Garrick BJ (1981) On the quantitative definition of risk. Risk Anal, 1(1):11–27
[Kum07] Kumamoto H (2007) Satisfying safety goals by probabilistic risk assessment. Reliability engineering. Springer
[KW17] Koopman P, Wagner M (2017) Autonomous vehicle safety: An interdisciplinary challenge. IEEE Int Transp Syst Mag, 9(1):90–96
[Lev95] Leveson NG (1995) Safeware: system safety and computers. Addison-Wesley
[Lev04] Leveson NG (2004) A new accident model for engineering safer systems. Safety Sci, 42(4):237–70
[Lev12] Leveson NG (2012) Engineering a safer world: systems thinking applied to safety. Engineering systems. MIT Press
[LFL13] Leitner-Fischer F, Leue S (2013) Probabilistic fault tree synthesis using causality computation. Int J Critical Comput-Based Syst, 4(2):119–43
[LS87] Leveson NG, Stolzy JL (1987) Safety analysis using Petri nets. IEEE Trans Softw Eng, 13(3):386–97
[LS09] Leucker M, Schallhart C (2009) A brief account of runtime verification. J Logic Algeb Program, 78(5):293–303
[LSS11] Lund MS, Solhaug B, Stølen K (2011) Model-driven risk analysis: The CORAS approach. Springer
[McD94] McDermid John A (1994) Support for safety cases and safety arguments using SAM. Reliability Engineering & System Safety, 43(2):111–127
[MGW+18] Machin M, Guiochet J, Waeselynck H, Blanquart J-P, Roy M, Masson L (2018) SMOF – a safety monitoring framework for autonomous systems. IEEE Trans Syst Man Cybern: Syst, 48(5):702–715
[MJG+11] Meredith PO, Jin D, Griffith D, Chen F, Roşu G (2011) An overview of the MOP runtime verification framework. Int J Softw Tools Technol Trans, 14(3):249–289
[MMBG+12] Mekki-Mokhtar A, Blanquart J-P, Guiochet J, Powell D, Roy M (2012) Safety trigger conditions for critical autonomous systems. In: Dependable Computing (PRDC), 18th IEEE Pacific Rim International symposium. IEEE
[MS14] Müller J, Sukhatme GS (2014) Risk-aware trajectory generation with application to safe quadrotor landing. In: Intelligent Robots and Systems (IROS), IEEE/RSJ International conference
[NSV03] Netravali AN, Sabnani KK, Viswanathan R (2003) Correct passive testing algorithms and complete fault coverage. In: Formal techniques for networked and distributed systems (FORTE), pp 303–318. Springer
[OCW09] Oliveira M, Cavalcanti A, Woodcock J (2009) A UTP semantics for *Circus*. Formal Aspects Comput, 21(1-2):3–32
[ORS06] Ortmeier F, Reif W, Schellhorn G (2006) Deductive cause-consequence analysis (DCCA). IFAC Proc Vol, 38(1):62–67
[PBHS13] Pereira AA, Binney J, Hollinger GA, Sukhatme GS (2013) Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. J Field Robot, 30(5):741–762
[Ros10] Roscoe AW (2010) Understanding concurrent systems. Springer
[San14] Sanger TD (2014) Risk-aware control. Neural Comput, 26(12):2669–2691
[SC88] Sobek RP, Chatila RG (1988) Integrated planning and execution control for an autonomous mobile robot. Artif Intell Eng, 3(2):103–113
[Sch99] Schneider S (1999) Concurrent and real-time systems: the CSP approach. Wiley, New York
[She03] Sheridan TB (2003) Telerobotics, automation, and human supervisory control. The MIT Press
[Sim94] Simmons RG (1994) Structured control for autonomous robots. IEEE Trans Robot Autom, 10(1):34–43
[SLJS16] Sorin A, Larsen M, Jensen K, Schultz UP (2016) Rule-based dynamic safety monitoring for mobile robots. J Softw Eng Robot, 7(1):120–141
[SR02] Svedung I, Rasmussen J (2002) Graphic representation of accident scenarios: mapping system structure and the causation of accidents. Safety Sci, 40(5):397–417
[SSSS18] Shalev-Shwartz S, Shammah S, Shashua A (2018) On a formal model of safe and scalable self-driving cars. Technical report, Mobileye
[Tre08] Tretmans J (2008) Model based testing with labelled transition systems. In: Formal methods and testing, pp 1–38. Springer
[Uni48] United Nations General Assembly (1948) Universal declaration of human rights. Technical report research 217 A (III), United Nations General Assembly. [hereinafter "UDHR"].
[UPM18] Unanue JIA, Papadopoulos Y, Merle G (2018) Explicit modelling and treatment of repair in prediction of dependability. IEEE Trans Depend Secure Comput, pp 1–16
[VJK16] Volk M, Junges S, Katoen J-P (2016) Advancing dynamic fault tree analysis – get succinct state spaces fast and synthesise failure rates. In: Computer safety, reliability, and security (SAFECOMP), 35th international conference, pp 253–265
[War12] Warburton N (2012) Philosophy: the basics. Taylor & Francis

Received 15 September 2020
Accepted in revised form 13 March 2021 by Cliff Jones